



**HAL**  
open science

# Loop freedom of the (untimed) AODV routing protocol

Timothy Bourke, Peter Höfner

► **To cite this version:**

Timothy Bourke, Peter Höfner. Loop freedom of the (untimed) AODV routing protocol. 2014, pp.496.  
hal-01104033

**HAL Id: hal-01104033**

**<https://inria.hal.science/hal-01104033v1>**

Submitted on 15 Jan 2015

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Loop freedom of the (untimed) AODV routing protocol

Timothy Bourke<sup>1</sup>

Peter Höfner<sup>2</sup>

November 3, 2014

<sup>1</sup>Inria, École normale supérieure, and NICTA

<sup>2</sup>NICTA and Computer Science and Engineering, UNSW

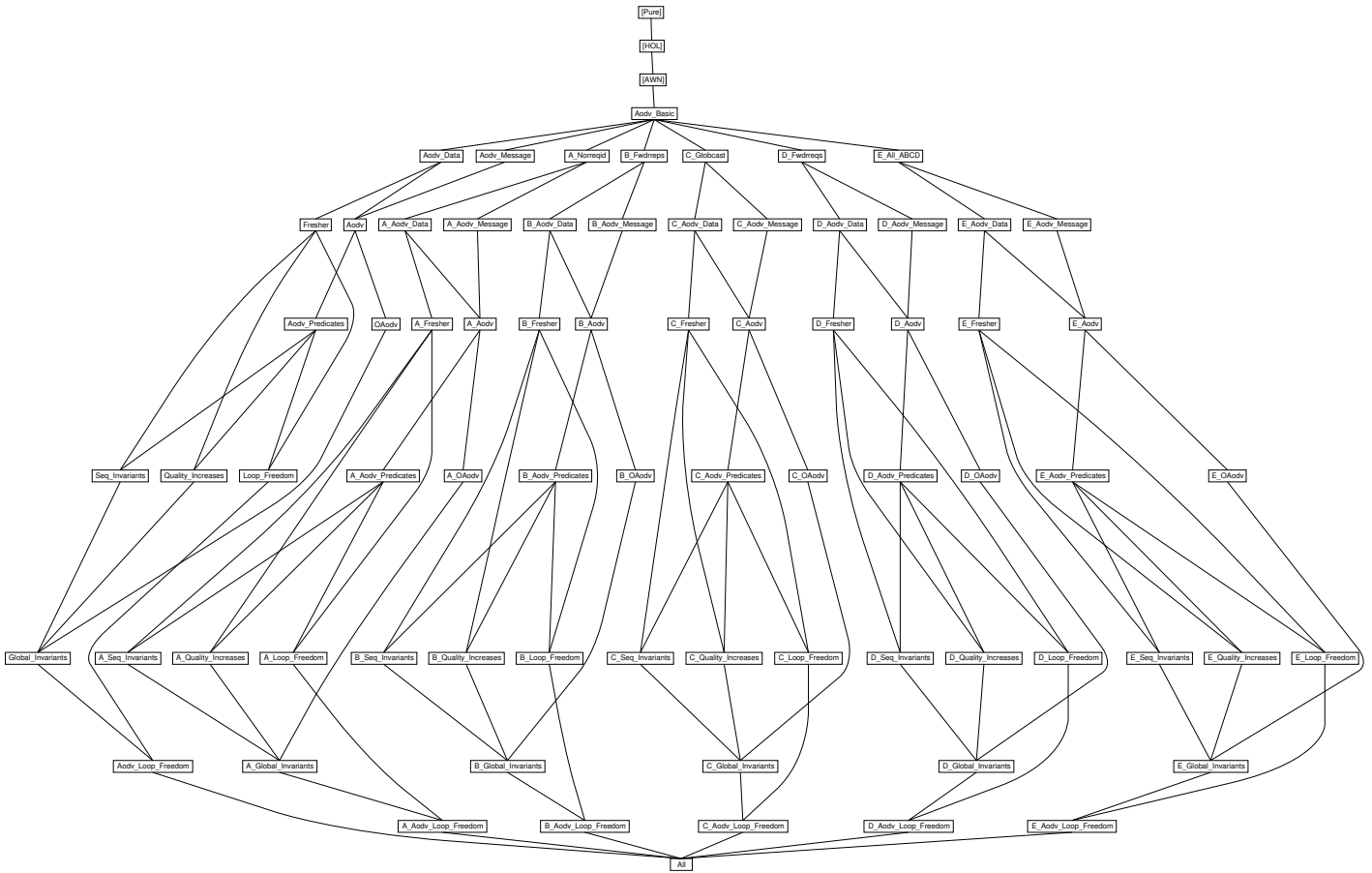
## Abstract

The Ad hoc On-demand Distance Vector (AODV) routing protocol [6] allows the nodes in a Mobile Ad hoc Network (MANET) or a Wireless Mesh Network (WMN) to know where to forward data packets. Such a protocol is ‘loop free’ if it never leads to routing decisions that forward packets in circles.

This development mechanises an existing pen-and-paper proof of loop freedom of AODV [4]. The protocol is modelled in the Algebra of Wireless Networks (AWN), which is the subject of an earlier paper [3] and mechanization [1]. The proof relies on a novel compositional approach for lifting invariants to networks of nodes.

We exploit the mechanization to analyse several variants of AODV and show that Isabelle/HOL can re-establish most proof obligations automatically and identify exactly the steps that are no longer valid. Each of the variants is essentially a modified copy of the main development.

Further documentation is available in [2].



# Contents

0.1	Basic data types and constants . . . . .	5
0.2	Predicates and functions used in the AODV model . . . . .	5
0.2.1	Sequence Numbers . . . . .	5
0.2.2	Modelling Routes . . . . .	6
0.2.3	Routing Tables . . . . .	6
0.2.4	Updating Routing Tables . . . . .	9
0.2.5	Route Requests . . . . .	19
0.2.6	Queued Packets . . . . .	19
0.2.7	Comparison with the original technical report . . . . .	20
0.3	AODV protocol messages . . . . .	20
0.4	The AODV protocol . . . . .	21
0.4.1	Data state . . . . .	22
0.4.2	Auxilliary message handling definitions . . . . .	23
0.4.3	The protocol process . . . . .	24
0.5	Invariant assumptions and properties . . . . .	30
0.6	Quality relations between routes . . . . .	32
0.6.1	Net sequence numbers . . . . .	32
0.6.2	Comparing routes . . . . .	34
0.6.3	Comparing routing tables . . . . .	35
0.6.4	Strictly comparing routing tables . . . . .	40
0.7	Invariant proofs on individual processes . . . . .	44
0.8	The quality increases predicate . . . . .	54
0.9	The ‘open’ AODV model . . . . .	61
0.10	Global invariant proofs over sequential processes . . . . .	61
0.11	Routing graphs and loop freedom . . . . .	79
0.12	Lift and transfer invariants to show loop freedom . . . . .	81
0.12.1	Lift to parallel processes with queues . . . . .	81
0.12.2	Lift to nodes . . . . .	83
0.12.3	Lift to partial networks . . . . .	84
0.12.4	Lift to closed networks . . . . .	84
0.12.5	Transfer into the standard model . . . . .	85
0.12.6	Loop freedom of AODV . . . . .	87
<b>1</b>	<b>Variant A: Skipping the RREQ ID</b> . . . . .	<b>88</b>
1.1	Predicates and functions used in the AODV model . . . . .	88
1.1.1	Sequence Numbers . . . . .	88
1.1.2	Modelling Routes . . . . .	88
1.1.3	Routing Tables . . . . .	89
1.1.4	Updating Routing Tables . . . . .	92
1.1.5	Route Requests . . . . .	101
1.1.6	Queued Packets . . . . .	102
1.1.7	Comparison with the original technical report . . . . .	103
1.2	AODV protocol messages . . . . .	103
1.3	The AODV protocol . . . . .	104
1.3.1	Data state . . . . .	104
1.3.2	Auxilliary message handling definitions . . . . .	105
1.3.3	The protocol process . . . . .	107

1.4	Invariant assumptions and properties . . . . .	112
1.5	Quality relations between routes . . . . .	114
1.5.1	Net sequence numbers . . . . .	114
1.5.2	Comparing routes . . . . .	117
1.5.3	Comparing routing tables . . . . .	117
1.5.4	Strictly comparing routing tables . . . . .	123
1.6	Invariant proofs on individual processes . . . . .	127
1.7	The quality increases predicate . . . . .	136
1.8	The ‘open’ AODV model . . . . .	143
1.9	Global invariant proofs over sequential processes . . . . .	144
1.10	Routing graphs and loop freedom . . . . .	162
1.11	Lift and transfer invariants to show loop freedom . . . . .	164
1.11.1	Lift to nodes . . . . .	165
1.11.2	Lift to partial networks . . . . .	166
1.11.3	Lift to closed networks . . . . .	167
1.11.4	Transfer into the standard model . . . . .	168
1.11.5	Loop freedom of AODV . . . . .	169
<b>2</b>	<b>Variant B: Forwarding the Route Reply</b> . . . . .	<b>170</b>
2.1	Predicates and functions used in the AODV model . . . . .	170
2.1.1	Sequence Numbers . . . . .	170
2.1.2	Modelling Routes . . . . .	170
2.1.3	Routing Tables . . . . .	171
2.1.4	Updating Routing Tables . . . . .	174
2.1.5	Route Requests . . . . .	184
2.1.6	Queued Packets . . . . .	184
2.1.7	Comparison with the original technical report . . . . .	185
2.2	AODV protocol messages . . . . .	185
2.3	The AODV protocol . . . . .	186
2.3.1	Data state . . . . .	186
2.3.2	Auxilliary message handling definitions . . . . .	188
2.3.3	The protocol process . . . . .	189
2.4	Invariant assumptions and properties . . . . .	195
2.5	Quality relations between routes . . . . .	197
2.5.1	Net sequence numbers . . . . .	197
2.5.2	Comparing routes . . . . .	199
2.5.3	Comparing routing tables . . . . .	200
2.5.4	Strictly comparing routing tables . . . . .	205
2.6	Invariant proofs on individual processes . . . . .	209
2.7	The quality increases predicate . . . . .	219
2.8	The ‘open’ AODV model . . . . .	226
2.9	Global invariant proofs over sequential processes . . . . .	227
2.10	Routing graphs and loop freedom . . . . .	245
2.11	Lift and transfer invariants to show loop freedom . . . . .	247
2.11.1	Lift to parallel processes with queues . . . . .	247
2.11.2	Lift to nodes . . . . .	248
2.11.3	Lift to partial networks . . . . .	249
2.11.4	Lift to closed networks . . . . .	250
2.11.5	Transfer into the standard model . . . . .	250
2.11.6	Loop freedom of AODV . . . . .	252
<b>3</b>	<b>Variant C: From Groupcast to Broadcast</b> . . . . .	<b>253</b>
3.1	Predicates and functions used in the AODV model . . . . .	253
3.1.1	Sequence Numbers . . . . .	253
3.1.2	Modelling Routes . . . . .	253
3.1.3	Routing Tables . . . . .	254
3.1.4	Updating Routing Tables . . . . .	257

3.1.5	Route Requests . . . . .	265
3.1.6	Queued Packets . . . . .	265
3.1.7	Comparison with the original technical report . . . . .	266
3.2	AODV protocol messages . . . . .	266
3.3	The AODV protocol . . . . .	267
3.3.1	Data state . . . . .	267
3.3.2	Auxilliary message handling definitions . . . . .	268
3.3.3	The protocol process . . . . .	270
3.4	Invariant assumptions and properties . . . . .	275
3.5	Quality relations between routes . . . . .	277
3.5.1	Net sequence numbers . . . . .	277
3.5.2	Comparing routes . . . . .	279
3.5.3	Comparing routing tables . . . . .	280
3.5.4	Strictly comparing routing tables . . . . .	285
3.6	Invariant proofs on individual processes . . . . .	289
3.7	The quality increases predicate . . . . .	298
3.8	The ‘open’ AODV model . . . . .	305
3.9	Global invariant proofs over sequential processes . . . . .	306
3.10	Routing graphs and loop freedom . . . . .	323
3.11	Lift and transfer invariants to show loop freedom . . . . .	325
3.11.1	Lift to parallel processes with queues . . . . .	325
3.11.2	Lift to nodes . . . . .	327
3.11.3	Lift to partial networks . . . . .	328
3.11.4	Lift to closed networks . . . . .	329
3.11.5	Transfer into the standard model . . . . .	329
3.11.6	Loop freedom of AODV . . . . .	331
<b>4</b>	<b>Variant D: Forwarding the Route Request</b> . . . . .	<b>332</b>
4.1	Predicates and functions used in the AODV model . . . . .	332
4.1.1	Sequence Numbers . . . . .	332
4.1.2	Modelling Routes . . . . .	332
4.1.3	Routing Tables . . . . .	333
4.1.4	Updating Routing Tables . . . . .	336
4.1.5	Route Requests . . . . .	346
4.1.6	Queued Packets . . . . .	346
4.1.7	Comparison with the original technical report . . . . .	347
4.2	AODV protocol messages . . . . .	347
4.3	The AODV protocol . . . . .	348
4.3.1	Data state . . . . .	348
4.3.2	Auxilliary message handling definitions . . . . .	350
4.3.3	The protocol process . . . . .	351
4.4	Invariant assumptions and properties . . . . .	357
4.5	Quality relations between routes . . . . .	359
4.5.1	Net sequence numbers . . . . .	359
4.5.2	Comparing routes . . . . .	361
4.5.3	Comparing routing tables . . . . .	362
4.5.4	Strictly comparing routing tables . . . . .	367
4.6	Invariant proofs on individual processes . . . . .	371
4.7	The quality increases predicate . . . . .	381
4.8	The ‘open’ AODV model . . . . .	388
4.9	Global invariant proofs over sequential processes . . . . .	389
4.10	Routing graphs and loop freedom . . . . .	407
4.11	Lift and transfer invariants to show loop freedom . . . . .	409
4.11.1	Lift to parallel processes with queues . . . . .	409
4.11.2	Lift to nodes . . . . .	410
4.11.3	Lift to partial networks . . . . .	411
4.11.4	Lift to closed networks . . . . .	412

4.11.5	Transfer into the standard model . . . . .	412
4.11.6	Loop freedom of AODV . . . . .	414
<b>5</b>	<b>Variants A–D: All proposed modifications</b>	<b>415</b>
5.1	Predicates and functions used in the AODV model . . . . .	415
5.1.1	Sequence Numbers . . . . .	415
5.1.2	Modelling Routes . . . . .	415
5.1.3	Routing Tables . . . . .	416
5.1.4	Updating Routing Tables . . . . .	418
5.1.5	Route Requests . . . . .	425
5.1.6	Queued Packets . . . . .	427
5.1.7	Comparison with the original technical report . . . . .	427
5.2	AODV protocol messages . . . . .	428
5.3	The AODV protocol . . . . .	429
5.3.1	Data state . . . . .	429
5.3.2	Auxilliary message handling definitions . . . . .	430
5.3.3	The protocol process . . . . .	432
5.4	Invariant assumptions and properties . . . . .	437
5.5	Quality relations between routes . . . . .	439
5.5.1	Net sequence numbers . . . . .	439
5.5.2	Comparing routes . . . . .	441
5.5.3	Comparing routing tables . . . . .	442
5.5.4	Strictly comparing routing tables . . . . .	447
5.6	Invariant proofs on individual processes . . . . .	451
5.7	The quality increases predicate . . . . .	460
5.8	The ‘open’ AODV model . . . . .	467
5.9	Global invariant proofs over sequential processes . . . . .	468
5.10	Routing graphs and loop freedom . . . . .	486
5.11	Lift and transfer invariants to show loop freedom . . . . .	488
5.11.1	Lift to parallel processes with queues . . . . .	488
5.11.2	Lift to nodes . . . . .	489
5.11.3	Lift to partial networks . . . . .	490
5.11.4	Lift to closed networks . . . . .	491
5.11.5	Transfer into the standard model . . . . .	491
5.11.6	Loop freedom of AODV . . . . .	493

## 0.1 Basic data types and constants

```
theory Aodv_Basic
imports Main "../AWN/AWN_SOS"
begin
```

These definitions are shared with all variants.

```
type_synonym rreqid = nat
type_synonym sqn = nat
```

```
datatype k = Known | Unknown
abbreviation kno where "kno  $\equiv$  Known"
abbreviation unk where "unk  $\equiv$  Unknown"
```

```
datatype p = NoRequestRequired | RequestRequired
abbreviation noreq where "noreq  $\equiv$  NoRequestRequired"
abbreviation req where "req  $\equiv$  RequestRequired"
```

```
datatype f = Valid | Invalid
abbreviation val where "val  $\equiv$  Valid"
abbreviation inv where "inv  $\equiv$  Invalid"
```

```
lemma not_ks [simp]:
  "(x  $\neq$  kno) = (x = unk)"
  "(x  $\neq$  unk) = (x = kno)"
  by (cases x, clarsimp+)
```

```
lemma not_ps [simp]:
  "(x  $\neq$  noreq) = (x = req)"
  "(x  $\neq$  req) = (x = noreq)"
  by (cases x, clarsimp+)
```

```
lemma not_ffs [simp]:
  "(x  $\neq$  val) = (x = inv)"
  "(x  $\neq$  inv) = (x = val)"
  by (cases x, clarsimp+)
```

end

## 0.2 Predicates and functions used in the AODV model

```
theory Aodv_Data
imports Aodv_Basic
begin
```

### 0.2.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

```
definition inc :: "sqn  $\Rightarrow$  sqn"
  where "inc sn  $\equiv$  if sn = 0 then sn else sn + 1"
```

```
lemma less_than_inc [simp]: "x  $\leq$  inc x"
  unfolding inc_def by simp
```

```
lemma inc_minus_suc_0 [simp]:
  "inc x - Suc 0 = x"
  unfolding inc_def by simp
```

```
lemma inc_never_one' [simp, intro]: "inc x  $\neq$  Suc 0"
  unfolding inc_def by simp
```

```
lemma inc_never_one [simp, intro]: "inc x  $\neq$  1"
  by simp
```



## 0.2.2 Modelling Routes

A route is a 6-tuple,  $(dsn, dsk, flag, hops, nhip, pre)$  where  $dsn$  is the ‘destination sequence number’,  $dsk$  is the ‘destination-sequence-number status’,  $flag$  is the route status,  $hops$  is the number of hops to the destination,  $nhip$  is the next hop toward the destination, and  $pre$  is the set of ‘precursor nodes’ those interested in hearing about changes to the route.

```
type_synonym r = "sqn × k × f × nat × ip × ip set"
```

```
definition proj2 :: "r ⇒ sqn" ("π2")
  where "π2 ≡ λ(dsn, _, _, _, _). dsn"
```

```
definition proj3 :: "r ⇒ k" ("π3")
  where "π3 ≡ λ(_, dsk, _, _, _). dsk"
```

```
definition proj4 :: "r ⇒ f" ("π4")
  where "π4 ≡ λ(_, _, flag, _, _, _). flag"
```

```
definition proj5 :: "r ⇒ nat" ("π5")
  where "π5 ≡ λ(_, _, _, hops, _, _). hops"
```

```
definition proj6 :: "r ⇒ ip" ("π6")
  where "π6 ≡ λ(_, _, _, _, nhip, _). nhip"
```

```
definition proj7 :: "r ⇒ ip set" ("π7")
  where "π7 ≡ λ(_, _, _, _, _, pre). pre"
```

```
lemma projs [simp]:
  "π2(dsn, dsk, flag, hops, nhip, pre) = dsn"
  "π3(dsn, dsk, flag, hops, nhip, pre) = dsk"
  "π4(dsn, dsk, flag, hops, nhip, pre) = flag"
  "π5(dsn, dsk, flag, hops, nhip, pre) = hops"
  "π6(dsn, dsk, flag, hops, nhip, pre) = nhip"
  "π7(dsn, dsk, flag, hops, nhip, pre) = pre"
  by (clarsimp simp: proj2_def proj3_def proj4_def
      proj5_def proj6_def proj7_def)+
```

```
lemma proj3_pred [intro]: "[[ P kno; P unk ] ⇒ P (π3 x)]"
  by (rule k.induct)
```

```
lemma proj4_pred [intro]: "[[ P val; P inv ] ⇒ P (π4 x)]"
  by (rule f.induct)
```

```
lemma proj6_pair_snd [simp]:
  fixes dsn' r
  shows "π6(dsn', snd(r)) = π6(r)"
  by (cases r) simp
```

## 0.2.3 Routing Tables

Routing tables map ip addresses to route entries.

```
type_synonym rt = "ip → r"
```

```
syntax
  "_Sigma_route" :: "rt ⇒ ip → r" ("σroute'(_, _)'")
```

```
translations
  "σroute(rt, dip)" => "rt dip"
```

```
definition sqn :: "rt ⇒ ip ⇒ sqn"
  where "sqn rt dip ≡ case σroute(rt, dip) of Some r ⇒ π2(r) | None ⇒ 0"
```

```
definition sqnf :: "rt ⇒ ip ⇒ k"
  where "sqnf rt dip ≡ case σroute(rt, dip) of Some r ⇒ π3(r) | None ⇒ unk"
```

```

abbreviation flag :: "rt ⇒ ip → f"
  where "flag rt dip ≡ map_option π4 (σroute(rt, dip))"

abbreviation dhops :: "rt ⇒ ip → nat"
  where "dhops rt dip ≡ map_option π5 (σroute(rt, dip))"

abbreviation nhop :: "rt ⇒ ip → ip"
  where "nhop rt dip ≡ map_option π6 (σroute(rt, dip))"

abbreviation precs :: "rt ⇒ ip → ip set"
  where "precs rt dip ≡ map_option π7 (σroute(rt, dip))"

definition vD :: "rt ⇒ ip set"
  where "vD rt ≡ {dip. flag rt dip = Some val}"

definition iD :: "rt ⇒ ip set"
  where "iD rt ≡ {dip. flag rt dip = Some inv}"

definition kD :: "rt ⇒ ip set"
  where "kD rt ≡ {dip. rt dip ≠ None}"

lemma kD_is_vD_and_iD: "kD rt = vD rt ∪ iD rt"
  unfolding kD_def vD_def iD_def by auto

lemma vD_iD_gives_kD [simp]:
  "∧ip rt. ip ∈ vD rt ⇒ ip ∈ kD rt"
  "∧ip rt. ip ∈ iD rt ⇒ ip ∈ kD rt"
  unfolding kD_is_vD_and_iD by simp_all

lemma kD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ kD rt"
  shows "∃ dsn dsk flag hops nhip pre.
    σroute(rt, dip) = Some (dsn, dsk, flag, hops, nhip, pre)"
  using assms unfolding kD_def by simp

lemma kD_None [dest]:
  fixes dip rt
  assumes "dip ∉ kD rt"
  shows "σroute(rt, dip) = None"
  using assms unfolding kD_def
  by (metis (mono_tags) mem_Collect_eq)

lemma vD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ vD rt"
  shows "∃ dsn dsk hops nhip pre.
    σroute(rt, dip) = Some (dsn, dsk, val, hops, nhip, pre)"
  using assms unfolding vD_def by simp

lemma vD_empty [simp]: "vD Map.empty = {}"
  unfolding vD_def by simp

lemma iD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ iD rt"
  shows "∃ dsn dsk hops nhip pre.
    σroute(rt, dip) = Some (dsn, dsk, inv, hops, nhip, pre)"
  using assms unfolding iD_def by simp

lemma val_is_vD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"

```

```

    and "the (flag rt ip) = val"
  shows "ip∈vD(rt)"
using assms unfolding vD_def by auto

lemma inv_is_iD [elim]:
  fixes ip rt
  assumes "ip∈kD(rt)"
    and "the (flag rt ip) = inv"
  shows "ip∈iD(rt)"
using assms unfolding iD_def by auto

lemma iD_flag_is_inv [elim, simp]:
  fixes ip rt
  assumes "ip∈iD(rt)"
  shows "the (flag rt ip) = inv"
proof -
  from ⟨ip∈iD(rt)⟩ have "ip∈kD(rt)" by auto
  with assms show ?thesis unfolding iD_def by auto
qed

lemma kD_but_not_vD_is_iD [elim]:
  fixes ip rt
  assumes "ip∈kD(rt)"
    and "ip∉vD(rt)"
  shows "ip∈iD(rt)"
proof -
  from ⟨ip∈kD(rt)⟩ obtain dsn dsk f hops nhop pre
    where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop, pre)"
    by (metis kD_Some)
  from ⟨ip∉vD(rt)⟩ have "f ≠ val"
  proof (rule contrapos_nn)
    assume "f = val"
    with rtip have "the (flag rt ip) = val" by simp
    with ⟨ip∈kD(rt)⟩ show "ip∈vD(rt)" ..
  qed
  with rtip have "the (flag rt ip) = inv" by simp
  with ⟨ip∈kD(rt)⟩ show "ip∈iD(rt)" ..
qed

lemma vD_or_iD [elim]:
  fixes ip rt
  assumes "ip∈kD(rt)"
    and "ip∈vD(rt) ⇒ P rt ip"
    and "ip∈iD(rt) ⇒ P rt ip"
  shows "P rt ip"
proof -
  from ⟨ip∈kD(rt)⟩ have "ip∈vD(rt) ∪ iD(rt)"
    by (simp add: kD_is_vD_and_iD)
  thus ?thesis by (auto elim: assms(2-3))
qed

lemma proj5_eq_dhops: "∧dip rt. dip∈kD(rt) ⇒ π5(the (rt dip)) = the (dhops rt dip)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj4_eq_flag: "∧dip rt. dip∈kD(rt) ⇒ π4(the (rt dip)) = the (flag rt dip)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj2_eq_sqn: "∧dip rt. dip∈kD(rt) ⇒ π2(the (rt dip)) = sqn rt dip"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 [simp]:
  "∧ip rt. ip∈kD(rt) ⇒ sqnf rt ip = π3(the (rt ip))"
  unfolding sqnf_def by auto

```

```
lemma vD_flag_val [simp]:
  " $\bigwedge \text{dip } rt. \text{dip} \in \text{vD } (rt) \implies \text{the } (\text{flag } rt \text{ dip}) = \text{val}$ "
  unfolding vD_def by clarsimp
```

```
lemma kD_update [simp]:
  " $\bigwedge rt \text{ nip } v. \text{kD } (rt(\text{nip} \mapsto v)) = \text{insert } \text{nip } (\text{kD } rt)$ "
  unfolding kD_def by auto
```

```
lemma kD_empty [simp]: "kD Map.empty = {}"
  unfolding kD_def by simp
```

```
lemma ip_equal_or_known [elim]:
  fixes rt ip ip'
  assumes "ip = ip'  $\vee$  ip  $\in$  kD(rt)"
    and "ip = ip'  $\implies$  P rt ip ip'"
    and " $\llbracket \text{ip} \neq \text{ip}'; \text{ip} \in \text{kD}(rt) \rrbracket \implies P \text{ rt ip ip}'$ "
  shows "P rt ip ip'"
  using assms by auto
```

## 0.2.4 Updating Routing Tables

Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.

### Updating Precursor Lists

```
definition addpre :: "r  $\Rightarrow$  ip set  $\Rightarrow$  r"
  where "addpre r npre  $\equiv$  let (dsn, dsk, flag, hops, nhip, pre) = r in
    (dsn, dsk, flag, hops, nhip, pre  $\cup$  npre)"
```

```
lemma proj2_addpre:
  fixes v pre
  shows " $\pi_2(\text{addpre } v \text{ pre}) = \pi_2(v)$ "
  unfolding addpre_def by (cases v) simp
```

```
lemma proj3_addpre:
  fixes v pre
  shows " $\pi_3(\text{addpre } v \text{ pre}) = \pi_3(v)$ "
  unfolding addpre_def by (cases v) simp
```

```
lemma proj4_addpre:
  fixes v pre
  shows " $\pi_4(\text{addpre } v \text{ pre}) = \pi_4(v)$ "
  unfolding addpre_def by (cases v) simp
```

```
lemma proj5_addpre:
  fixes v pre
  shows " $\pi_5(\text{addpre } v \text{ pre}) = \pi_5(v)$ "
  unfolding addpre_def by (cases v) simp
```

```
lemma proj6_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows " $\pi_6(\text{addpre } v \text{ npre}) = \pi_6(v)$ "
  unfolding addpre_def by (cases v) simp
```

```
lemma proj7_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows " $\pi_7(\text{addpre } v \text{ npre}) = \pi_7(v) \cup \text{npre}$ "
  unfolding addpre_def by (cases v) simp
```

```
lemma addpre_empty: "addpre r {} = r"
  unfolding addpre_def by simp
```

```
lemma addpre_r:
```

```

"addpre (dsn, dsk, fl, hops, nhop, pre) npre = (dsn, dsk, fl, hops, nhop, pre  $\cup$  npre)"
unfolding addpre_def by simp

lemmas addpre_simps [simp] = proj2_addpre proj3_addpre proj4_addpre proj5_addpre
proj6_addpre proj7_addpre addpre_empty addpre_r

definition addpreRT :: "rt  $\Rightarrow$  ip  $\Rightarrow$  ip set  $\rightarrow$  rt"
where "addpreRT rt dip npre  $\equiv$ 
map_option ( $\lambda$ s. rt (dip  $\mapsto$  addpre s npre)) ( $\sigma_{route}(rt, dip)$ )"

lemma snd_addpre [simp]:
" $\wedge$ dsn dsn' v pre. (dsn, snd(addpre (dsn', v) pre)) = addpre (dsn, v) pre"
unfolding addpre_def by clarsimp

lemma proj2_addpreRT [simp]:
fixes ip rt ip' npre
assumes "ip  $\in$  kD rt"
and "ip'  $\in$  kD rt"
shows " $\pi_2(\text{the}(\text{the}(\text{addpreRT rt ip' npre}) ip)) = \pi_2(\text{the}(rt ip))$ "
using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj3_addpreRT [simp]:
fixes ip rt ip' npre
assumes "ip  $\in$  kD rt"
and "ip'  $\in$  kD rt"
shows " $\pi_3(\text{the}(\text{the}(\text{addpreRT rt ip' npre}) ip)) = \pi_3(\text{the}(rt ip))$ "
using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj5_addpreRT [simp]:
" $\wedge$ rt dip ip npre. dip  $\in$  kD(rt)  $\implies$   $\pi_5(\text{the}(\text{the}(\text{addpreRT rt dip npre}) ip)) = \pi_5(\text{the}(rt ip))$ "
unfolding addpreRT_def by auto

lemma flag_addpreRT [simp]:
fixes rt pre ip dip
assumes "dip  $\in$  kD rt"
shows "flag (the (addpreRT rt dip pre)) ip = flag rt ip"
unfolding addpreRT_def
using assms [THEN kD_Some] by (clarsimp)

lemma kD_addpreRT [simp]:
fixes rt dip npre
assumes "dip  $\in$  kD rt"
shows "kD (the (addpreRT rt dip npre)) = kD rt"
unfolding kD_def addpreRT_def
using assms [THEN kD_Some]
by clarsimp blast

lemma vD_addpreRT [simp]:
fixes rt dip npre
assumes "dip  $\in$  kD rt"
shows "vD (the (addpreRT rt dip npre)) = vD rt"
unfolding vD_def addpreRT_def
using assms [THEN kD_Some] by clarsimp auto

lemma iD_addpreRT [simp]:
fixes rt dip npre
assumes "dip  $\in$  kD rt"
shows "iD (the (addpreRT rt dip npre)) = iD rt"
unfolding iD_def addpreRT_def
using assms [THEN kD_Some] by clarsimp auto

lemma nhop_addpreRT [simp]:
fixes rt pre ip dip
assumes "dip  $\in$  kD rt"

```

```

  shows "nhop (the (addpreRT rt dip pre)) ip = nhop rt ip"
  unfolding sqn_def addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

```

```

lemma sqn_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "sqn (the (addpreRT rt dip pre)) ip = sqn rt ip"
  unfolding sqn_def addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

```

```

lemma dhops_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "dhops (the (addpreRT rt dip pre)) ip = dhops rt ip"
  unfolding addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

```

```

lemma sqnf_addpreRT [simp]:
  "∧ ip dip. ip ∈ kD(rt ξ) ⇒ sqnf (the (addpreRT (rt ξ) ip npre)) dip = sqnf (rt ξ) dip"
  unfolding sqnf_def addpreRT_def by auto

```

## Updating route entries

```

lemma in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∈ kD(rt)"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = es (the (rt dip))"
  using assms [THEN kD_Some] by auto

```

```

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∉ kD(rt)"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = en"
  using assms [THEN kD_None] by auto

```

```

lemma rt_Some_sqn [dest]:
  fixes rt and ip dsn dsk flag hops nhip pre
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows "sqn rt ip = dsn"
  unfolding sqn_def using assms by simp

```

```

lemma not_kD_sqn [simp]:
  fixes dip rt
  assumes "dip ∉ kD(rt)"
  shows "sqn rt dip = 0"
  using assms unfolding sqn_def
  by simp

```

```

definition update_arg_wf :: "r ⇒ bool"
where "update_arg_wf r ≡ π4(r) = val ∧
      (π2(r) = 0) = (π3(r) = unk) ∧
      (π3(r) = unk → π5(r) = 1)"

```

```

lemma update_arg_wf_gives_cases:
  "∧ r. update_arg_wf r ⇒ (π2(r) = 0) = (π3(r) = unk)"
  unfolding update_arg_wf_def by simp

```

```

lemma update_arg_wf_tuples [simp]:
  "∧ nhip pre. update_arg_wf (0, unk, val, Suc 0, nhip, pre)"
  "∧ n hops nhip pre. update_arg_wf (Suc n, kno, val, hops, nhip, pre)"
  unfolding update_arg_wf_def by auto

```

```

lemma update_arg_wf_tuples' [elim]:

```

" $\bigwedge n \text{ hops } nhip \text{ pre. Suc } 0 \leq n \implies \text{update\_arg\_wf } (n, \text{kno}, \text{val}, \text{hops}, nhip, \text{pre})"$   
 unfolding update\_arg\_wf\_def by auto

lemma wf\_r\_cases [intro]:

fixes P r  
 assumes "update\_arg\_wf r"  
 and c1: " $\bigwedge nhip \text{ pre. } P (0, \text{unk}, \text{val}, \text{Suc } 0, nhip, \text{pre})"$   
 and c2: " $\bigwedge \text{dsn } \text{hops } nhip \text{ pre. } \text{dsn} > 0 \implies P (\text{dsn}, \text{kno}, \text{val}, \text{hops}, nhip, \text{pre})"$   
 shows "P r"  
 proof -  
 obtain dsn dsk flag hops nhip pre  
 where \*: "r = (dsn, dsk, flag, hops, nhip, pre)" by (cases r)  
 with ⟨update\_arg\_wf r⟩ have wf1: "flag = val"  
 and wf2: "(dsn = 0) = (dsk = unk)"  
 and wf3: "dsk = unk  $\longrightarrow$  (hops = 1)"  
 unfolding update\_arg\_wf\_def by auto  
 have "P (dsn, dsk, flag, hops, nhip, pre)"  
 proof (cases dsk)  
 assume "dsk = unk"  
 moreover with wf2 wf3 have "dsn = 0" and "hops = Suc 0" by auto  
 ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c1)  
 next  
 assume "dsk = kno"  
 moreover with wf2 have "dsn > 0" by simp  
 ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c2)  
 qed  
 with \* show "P r" by simp  
 qed

definition update :: "rt  $\Rightarrow$  ip  $\Rightarrow$  r  $\Rightarrow$  rt"

where

"update rt ip r  $\equiv$   
 case  $\sigma_{\text{route}}(\text{rt}, \text{ip})$  of  
 None  $\Rightarrow$  rt (ip  $\mapsto$  r)  
 | Some s  $\Rightarrow$   
 if  $\pi_2(s) < \pi_2(r)$  then rt (ip  $\mapsto$  addpre r ( $\pi_7(s)$ ))  
 else if  $\pi_2(s) = \pi_2(r) \wedge (\pi_5(s) > \pi_5(r) \vee \pi_4(s) = \text{inv})$   
 then rt (ip  $\mapsto$  addpre r ( $\pi_7(s)$ ))  
 else if  $\pi_3(r) = \text{unk}$   
 then rt (ip  $\mapsto$  ( $\pi_2(s)$ , snd (addpre r ( $\pi_7(s)$ ))))  
 else rt (ip  $\mapsto$  addpre s ( $\pi_7(r)$ ))"

lemma update\_simps [simp]:

fixes r s nrt nr nr' ns rt ip  
 defines "s  $\equiv$  the  $\sigma_{\text{route}}(\text{rt}, \text{ip})"$   
 and "nr  $\equiv$  addpre r ( $\pi_7(s)$ )"  
 and "nr'  $\equiv$  ( $\pi_2(s)$ ,  $\pi_3(nr)$ ,  $\pi_4(nr)$ ,  $\pi_5(nr)$ ,  $\pi_6(nr)$ ,  $\pi_7(nr)$ )"  
 and "ns  $\equiv$  addpre s ( $\pi_7(r)$ )"  
 shows  
 "[ip  $\notin$  kD(rt)]  $\implies$  update rt ip r = rt (ip  $\mapsto$  r)"  
 "[ip  $\in$  kD(rt); sqn rt ip <  $\pi_2(r)$ ]  $\implies$  update rt ip r = rt (ip  $\mapsto$  nr)"  
 "[ip  $\in$  kD(rt); sqn rt ip =  $\pi_2(r)$ ;  
 the (dhops rt ip) >  $\pi_5(r)$ ]  $\implies$  update rt ip r = rt (ip  $\mapsto$  nr)"  
 "[ip  $\in$  kD(rt); sqn rt ip =  $\pi_2(r)$ ;  
 flag rt ip = Some inv]  $\implies$  update rt ip r = rt (ip  $\mapsto$  nr)"  
 "[ip  $\in$  kD(rt);  $\pi_3(r) = \text{unk}$ ; ( $\pi_2(r) = 0$ ) = ( $\pi_3(r) = \text{unk}$ )]  $\implies$  update rt ip r = rt (ip  $\mapsto$  nr)"  
 "[ip  $\in$  kD(rt); sqn rt ip  $\geq$   $\pi_2(r)$ ;  $\pi_3(r) = \text{kno}$ ;  
 sqn rt ip =  $\pi_2(r)$ ]  $\implies$  the (dhops rt ip)  $\leq$   $\pi_5(r) \wedge$  the (flag rt ip) = val ]  
 $\implies$  update rt ip r = rt (ip  $\mapsto$  ns)"

proof -

assume "ip  $\notin$  kD(rt)"  
 hence " $\sigma_{\text{route}}(\text{rt}, \text{ip}) = \text{None}$ " ..  
 thus "update rt ip r = rt (ip  $\mapsto$  r)"  
 unfolding update\_def by simp

```

next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip < π2(r)"
  from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
  with ⟨sqn rt ip < π2(r)⟩ show "update rt ip r = rt (ip ↦ nr)"
  unfolding update_def nr_def s_def by auto
next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip = π2(r)"
  and "the (dhops rt ip) > π5(r)"
  from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
  with ⟨sqn rt ip = π2(r)⟩ and ⟨the (dhops rt ip) > π5(r)⟩
  show "update rt ip r = rt (ip ↦ nr)"
  unfolding update_def nr_def s_def by auto
next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip = π2(r)"
  and "flag rt ip = Some inv"
  from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
  with ⟨sqn rt ip = π2(r)⟩ and ⟨flag rt ip = Some inv⟩
  show "update rt ip r = rt (ip ↦ nr)"
  unfolding update_def nr_def s_def by auto
next
  assume "ip ∈ kD(rt)"
  and "π3(r) = unk"
  and "(π2(r) = 0) = (π3(r) = unk)"
  from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
  with ⟨(π2(r) = 0) = (π3(r) = unk)⟩ and ⟨π3(r) = unk⟩
  show "update rt ip r = rt (ip ↦ nr)"
  unfolding update_def nr'_def nr_def s_def
  by (cases r) simp
next
  assume "ip ∈ kD(rt)"
  and otherassms: "sqn rt ip ≥ π2(r)"
  "π3(r) = kno"
  "sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val"
  from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
  with otherassms show "update rt ip r = rt (ip ↦ ns)"
  unfolding update_def ns_def s_def by auto
qed

```

lemma update\_cases [elim]:

```

assumes "(π2(r) = 0) = (π3(r) = unk)"
  and c1: "⟦ip ∉ kD(rt)⟧ ⇒ P (rt (ip ↦ r))"

  and c2: "⟦ip ∈ kD(rt); sqn rt ip < π2(r)⟧
    ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
  and c3: "⟦ip ∈ kD(rt); sqn rt ip = π2(r); the (dhops rt ip) > π5(r)⟧
    ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
  and c4: "⟦ip ∈ kD(rt); sqn rt ip = π2(r); the (flag rt ip) = inv⟧
    ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
  and c5: "⟦ip ∈ kD(rt); π3(r) = unk⟧
    ⇒ P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r),
      π4(r), π5(r), π6(r), π7(addpre r (π7(the σroute(rt, ip))))))"

```



```

and c6: "[[ip ∈ kD(rt); sqn rt ip ≥ π2(r); π3(r) = kno;
sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val]]
⇒ P (rt (ip ↦ addpre (the σroute(rt, ip)) (π7(r))))]"
shows "(P (update rt ip r))"
proof (cases "ip ∈ kD(rt)")
  assume "ip ∉ kD(rt)"
  with c1 show ?thesis
    by simp
next
assume "ip ∈ kD(rt)"
moreover then obtain dsn dsk fl hops nhip pre
  where rteq: "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
moreover obtain dsn' dsk' fl' hops' nhip' pre'
  where req: "r = (dsn', dsk', fl', hops', nhip', pre')"
  by (cases r) metis
ultimately show ?thesis
  using ⟨(π2(r) = 0) = (π3(r) = unk)⟩
  c2 [OF ⟨ip ∈ kD(rt)⟩]
  c3 [OF ⟨ip ∈ kD(rt)⟩]
  c4 [OF ⟨ip ∈ kD(rt)⟩]
  c5 [OF ⟨ip ∈ kD(rt)⟩]
  c6 [OF ⟨ip ∈ kD(rt)⟩]
  unfolding update_def sqn_def by auto
qed

```

lemma update\_cases\_kD:

```

assumes "(π2(r) = 0) = (π3(r) = unk)"
and "ip ∈ kD(rt)"
and c2: "sqn rt ip < π2(r) ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
and c3: "[[sqn rt ip = π2(r); the (dhops rt ip) > π5(r)]
⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip))))]"
and c4: "[[sqn rt ip = π2(r); the (flag rt ip) = inv]
⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip))))]"
and c5: "π3(r) = unk ⇒ P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r),
π4(r), π5(r), π6(r),
π7(addpre r (π7(the σroute(rt, ip))))))]"
and c6: "[[sqn rt ip ≥ π2(r); π3(r) = kno;
sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val]]
⇒ P (rt (ip ↦ addpre (the σroute(rt, ip)) (π7(r))))]"
shows "(P (update rt ip r))"
using assms(1) proof (rule update_cases)
  assume "sqn rt ip < π2(r)"
  thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))" by (rule c2)
next
  assume "sqn rt ip = π2(r)"
  and "the (dhops rt ip) > π5(r)"
  thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))"
  by (rule c3)
next
  assume "sqn rt ip = π2(r)"
  and "the (flag rt ip) = inv"
  thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))"
  by (rule c4)
next
  assume "π3(r) = unk"
  thus "P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r), π4(r), π5(r), π6(r),
π7(addpre r (π7(the (rt ip))))))]"
  by (rule c5)
next
  assume "sqn rt ip ≥ π2(r)"
  and "π3(r) = kno"
  and "sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val"
  thus "P (rt (ip ↦ addpre (the (rt ip)) (π7(r))))"

```

```

    by (rule c6)
qed (simp add: ⟨ip ∈ kD(rt)⟩)

lemma in_kD_after_update [simp]:
  fixes rt nip dsn dsk flag hops nhip pre
  shows "kD (update rt nip (dsn, dsk, flag, hops, nhip, pre)) = insert nip (kD rt)"
  unfolding update_def
  by (cases "rt nip") auto

lemma nhop_of_update [simp]:
  fixes rt dip dsn dsk flag hops nhip
  assumes "rt ≠ update rt dip (dsn, dsk, flag, hops, nhip, {})"
  shows "the (nhop (update rt dip (dsn, dsk, flag, hops, nhip, {})) dip) = nhip"
  proof -
  from assms
  have update_neq: "∧v. rt dip = Some v ⇒
    update rt dip (dsn, dsk, flag, hops, nhip, {})
    ≠ rt(dip ↦ addpre (the (rt dip)) (π7 (dsn, dsk, flag, hops, nhip, {})))"
  by auto
  show ?thesis
  proof (cases "rt dip = None")
  assume "rt dip = None"
  thus "?thesis" unfolding update_def by clarsimp
  next
  assume "rt dip ≠ None"
  then obtain v where "rt dip = Some v" by (metis not_None_eq)
  with update_neq [OF this] show ?thesis
  unfolding update_def by auto
  qed
qed

lemma sqn_if_updated:
  fixes rip v rt ip
  shows "sqn (λx. if x = rip then Some v else rt x) ip
    = (if ip = rip then π2(v) else sqn rt ip)"
  unfolding sqn_def by simp

lemma update_sqn [simp]:
  fixes rt dip rip dsn dsk hops nhip pre
  assumes "(dsn = 0) = (dsk = unk)"
  shows "sqn rt dip ≤ sqn (update rt rip (dsn, dsk, val, hops, nhip, pre)) dip"
  proof (rule update_cases)
  show "(π2 (dsn, dsk, val, hops, nhip, pre) = 0) = (π3 (dsn, dsk, val, hops, nhip, pre) = unk)"
  by simp (rule assms)
  qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
  fixes rt ip ip' dsn dsk flag hops nhip pre
  assumes "1 ≤ hops"
  shows "sqn rt ip ≤ sqn (update rt ip' (dsn, dsk, flag, hops, nhip, pre)) ip"
  using assms unfolding update_def sqn_def
  by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
  fixes rt dsn dsk flag hops ip rip nhip pre
  assumes ex: "∀ip∈kD rt. the (dhops rt ip) ≥ 1"
  and ip: "(ip = rip ∧ Suc 0 ≤ hops) ∨ (ip ≠ rip ∧ ip∈kD rt)"
  shows "Suc 0 ≤ the (dhops (update rt rip (dsn, dsk, flag, hops, nhip, pre)) ip)"
  using ip proof
  assume "ip = rip ∧ Suc 0 ≤ hops" thus ?thesis
  unfolding update_def using ex
  by (cases "rip ∈ kD rt") (drule(1) bspec, auto)
  next
  assume "ip ≠ rip ∧ ip∈kD rt" thus ?thesis

```

```

    using ex unfolding update_def
    by (cases "rip∈kD rt") auto
qed

lemma update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "(update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma nhop_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = nhop rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma dhops_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = dhops rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
  "∧rt ip dsn dsk flag hops nhip pre. sqn (rt(ip ↦ v)) ip = π2(v)"
  unfolding sqn_def by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip
  assumes "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nhip, {})) dip) = hops"
  using assms unfolding update_def
  by (clarsimp split: option.split_asm option.split split_if_asm) auto

lemma nhop_update_unk_val [simp]:
  "∧rt dip ip dsn hops npre.
  the (nhop (update rt dip (dsn, unk, val, hops, ip, npre)) dip) = ip"
  unfolding update_def by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
  assumes "update rt dip (dsn, dsk, flg, hops, sip, {}) ≠ rt"
  shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
  using assms unfolding update_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma update_rt_split_asm:
  "∧rt ip dsn dsk flag hops sip.
  P (update rt ip (dsn, dsk, flag, hops, sip, {}))
  =
  (¬(rt = update rt ip (dsn, dsk, flag, hops, sip, {})) ∧ ¬P rt
  ∨ rt ≠ update rt ip (dsn, dsk, flag, hops, sip, {})
  ∧ ¬P (update rt ip (dsn, dsk, flag, hops, sip, {})))"
  by auto

lemma sqn_update [simp]: "∧rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
  ⇒ sqn (update rt dip (dsn, kno, flg, hops, sip, {})) dip = dsn"
  unfolding update_def by (clarsimp split: option.split split_if_asm) auto

lemma sqnf_update [simp]: "∧rt dip dsn dsk flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})"

```

```

⇒ sqnf (update rt dip (dsn, dsk, flg, hops, sip, {})) dip = dsk"
unfolding update_def sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

lemma update_kno_dsn_greater_zero:
"∧rt dip ip dsn hops npre. 1 ≤ dsn ⇒ 1 ≤ (sqn (update rt dip (dsn, kno, val, hops, ip, npre)) dip)"
unfolding update_def
by (clarsimp split: option.splits)

lemma proj3_update [simp]: "∧rt dip dsn dsk flg hops sip.
rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
⇒ π3(the (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = dsk"
unfolding update_def sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

lemma nhop_update_changed_kno_val [simp]: "∧rt ip dsn dsk hops nhip.
rt ≠ update rt ip (dsn, kno, val, hops, nhip, {})
⇒ the (nhop (update rt ip (dsn, kno, val, hops, nhip, {})) ip) = nhip"
using assms unfolding update_def
by (clarsimp split: option.split_asm option.split split_if_asm) auto

lemma flag_update [simp]: "∧rt dip dsn flg hops sip.
rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
⇒ the (flag (update rt dip (dsn, kno, flg, hops, sip, {})) dip) = flg"
unfolding update_def
by (clarsimp split: option.split split_if_asm) auto

lemma the_flag_Some [dest!]:
fixes ip rt
assumes "the (flag rt ip) = x"
and "ip ∈ kD rt"
shows "flag rt ip = Some x"
using assms by auto

lemma kD_update_unchanged [dest]:
fixes rt dip dsn dsk flag hops nhip pre
assumes "rt = update rt dip (dsn, dsk, flag, hops, nhip, pre)"
shows "dip ∈ kD(rt)"
proof -
have "dip ∈ kD(update rt dip (dsn, dsk, flag, hops, nhip, pre))" by simp
with assms show ?thesis by simp
qed

lemma nhop_update [simp]: "∧rt dip dsn dsk flg hops sip.
rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
⇒ the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
unfolding update_def sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

lemma sqn_update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip pre
assumes "ip ≠ dip"
shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqn rt ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.splits) auto

lemma sqnf_update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip pre
assumes "ip ≠ dip"
shows "sqnf (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqnf rt ip"
using assms unfolding update_def sqnf_def
by (clarsimp split: option.splits) auto

lemma vD_update_val [dest]:

```

```
"^dip rt dip' dsn dsk hops nhip pre.
dip ∈ vD(update rt dip' (dsn, dsk, val, hops, nhip, pre)) ⇒ (dip∈vD(rt) ∨ dip=dip')"
unfolding update_def vD_def by (clarsimp split: option.split_asm split_if_asm)
```

## Invalidating route entries

```
definition invalidate :: "rt ⇒ (ip → sqn) ⇒ rt"
```

```
where "invalidate rt dests ≡
```

```
λip. case (rt ip, dests ip) of
  (None, _) ⇒ None
| (Some s, None) ⇒ Some s
| (Some (_, dsk, _, hops, nhip, pre), Some rsn) ⇒
  Some (rsn, dsk, inv, hops, nhip, pre)"
```

```
lemma proj3_invalidate [simp]:
```

```
"^dip. π3(the ((invalidate rt dests) dip)) = π3(the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)
```

```
lemma proj5_invalidate [simp]:
```

```
"^dip. π5(the ((invalidate rt dests) dip)) = π5(the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)
```

```
lemma proj6_invalidate [simp]:
```

```
"^dip. π6(the ((invalidate rt dests) dip)) = π6(the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)
```

```
lemma proj7_invalidate [simp]:
```

```
"^dip. π7(the ((invalidate rt dests) dip)) = π7(the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)
```

```
lemma invalidate_kD_inv [simp]:
```

```
"^rt dests. kD (invalidate rt dests) = kD rt"
unfolding invalidate_def kD_def
by (simp split: option.split)
```

```
lemma invalidate_sqn:
```

```
fixes rt dip dests
assumes "∀rsn. dests dip = Some rsn → sqn rt dip ≤ rsn"
shows "sqn rt dip ≤ sqn (invalidate rt dests) dip"
proof (cases "dip ∈ kD(rt)")
  assume "¬ dip ∈ kD(rt)"
  hence "dip∈kD(rt)" by simp
  then obtain dsn dsk flag hops nhip pre where "rt dip = Some (dsn, dsk, flag, hops, nhip, pre)"
    by (metis kD_Some)
  with assms show "sqn rt dip ≤ sqn (invalidate rt dests) dip"
    by (cases "dests dip") (auto simp add: invalidate_def sqn_def)
qed simp
```

```
lemma sqn_invalidate_in_dests [simp]:
```

```
fixes dests ipa rsn rt
assumes "dests ipa = Some rsn"
and "ipa∈kD(rt)"
shows "sqn (invalidate rt dests) ipa = rsn"
unfolding invalidate_def sqn_def
using assms(1) assms(2) [THEN kD_Some]
by clarsimp
```

```
lemma dhops_invalidate [simp]:
```

```
"^dip. the (dhops (invalidate rt dests) dip) = the (dhops rt dip)"
unfolding invalidate_def by (clarsimp split: option.split)
```

```
lemma sqnf_invalidate [simp]:
```

```
"^dip. sqnf (invalidate (rt ξ) (dests ξ)) dip = sqnf (rt ξ) dip"
unfolding sqnf_def invalidate_def by (clarsimp split: option.split)
```

```

lemma nhop_invalidate [simp]:
  "∧dip. the (nhop (invalidate (rt ξ) (dests ξ)) dip) = the (nhop (rt ξ) dip)"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
  fixes rt dests dip
  assumes "dip ∉ dom(dests)"
  shows "invalidate rt dests dip = rt dip"
  using assms unfolding invalidate_def
  by (clarsimp split: option.split_asm)

lemma invalidate_none [simp]:
  fixes rt dests dip
  assumes "dip ∉ kD(rt)"
  shows "invalidate rt dests dip = None"
  using assms unfolding invalidate_def by clarsimp

lemma vD_invalidate_vD_not_dests:
  "∧dip rt dests. dip ∈ vD(invalidate rt dests) ⇒ dip ∈ vD(rt) ∧ dests dip = None"
  unfolding invalidate_def vD_def
  by (clarsimp split: option.split_asm)

lemma sqn_invalidate_not_in_dests [simp]:
  fixes dests dip rt
  assumes "dip ∉ dom(dests)"
  shows "sqn (invalidate rt dests) dip = sqn rt dip"
  using assms unfolding sqn_def by simp

lemma invalidate_changes:
  fixes rt dests dip dsn dsk flag hops nhip pre
  assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows " dsn = (case dests dip of None ⇒ π2(the (rt dip)) | Some rsn ⇒ rsn)
    ∧ dsk = π3(the (rt dip))
    ∧ flag = (if dests dip = None then π4(the (rt dip)) else inv)
    ∧ hops = π5(the (rt dip))
    ∧ nhip = π6(the (rt dip))
    ∧ pre = π7(the (rt dip))"
  using assms unfolding invalidate_def
  by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: "∧dip rt dests. dip ∈ kD (rt)
  ⇒ π3(the (invalidate rt dests dip)) = π3(the (rt dip))"
  by (clarsimp simp: invalidate_def kD_def split: option.split)

lemma dests_iD_invalidate [simp]:
  assumes "dests ip = Some rsn"
  and "ip ∈ kD(rt)"
  shows "ip ∈ iD(invalidate rt dests)"
  using assms(1) assms(2) [THEN kD_Some] unfolding invalidate_def iD_def
  by (clarsimp split: option.split)

```

## 0.2.5 Route Requests

Generate a fresh route request identifier.

```

definition nrreqid :: "(ip × rreqid) set ⇒ ip ⇒ rreqid"
  where "nrreqid rreqs ip ≡ Max ({n. (ip, n) ∈ rreqs} ∪ {0}) + 1"

```

## 0.2.6 Queued Packets

Functions for sending data packets.

```

type-synonym store = "ip → (p × data list)"

```

```

definition sigma_queue :: "store  $\Rightarrow$  ip  $\Rightarrow$  data list" ("sigma_queue'(_, _)")
  where "sigma_queue(store, dip)  $\equiv$  case store dip of None  $\Rightarrow$  [] | Some (p, q)  $\Rightarrow$  q"

```

```

definition qD :: "store  $\Rightarrow$  ip set"
  where "qD  $\equiv$  dom"

```

```

definition add :: "data  $\Rightarrow$  ip  $\Rightarrow$  store  $\Rightarrow$  store"
  where "add d dip store  $\equiv$  case store dip of
    None  $\Rightarrow$  store (dip  $\mapsto$  (req, [d]))
    | Some (p, q)  $\Rightarrow$  store (dip  $\mapsto$  (p, q @ [d]))"

```

```

lemma qD_add [simp]:
  fixes d dip store
  shows "qD(add d dip store) = insert dip (qD store)"
  unfolding add_def Let_def qD_def
  by (clarsimp split: option.split)

```

```

definition drop :: "ip  $\Rightarrow$  store  $\rightarrow$  store"
  where "drop dip store  $\equiv$ 
    map_option ( $\lambda$ (p, q). if tl q = [] then store (dip := None)
      else store (dip  $\mapsto$  (p, tl q))) (store dip)"

```

```

definition sigma_p_flag :: "store  $\Rightarrow$  ip  $\rightarrow$  p" ("sigma_p_flag'(_, _)")
  where "sigma_p_flag(store, dip)  $\equiv$  map_option fst (store dip)"

```

```

definition unsetRRF :: "store  $\Rightarrow$  ip  $\Rightarrow$  store"
  where "unsetRRF store dip  $\equiv$  case store dip of
    None  $\Rightarrow$  store
    | Some (p, q)  $\Rightarrow$  store (dip  $\mapsto$  (noreq, q))"

```

```

definition setRRF :: "store  $\Rightarrow$  (ip  $\rightarrow$  sqn)  $\Rightarrow$  store"
  where "setRRF store dests  $\equiv$   $\lambda$ dip. if dests dip = None then store dip
    else map_option ( $\lambda$ (_, q). (req, q)) (store dip)"

```

## 0.2.7 Comparison with the original technical report

The major differences with the AODV technical report of Fehnker et al are:

1. *nhop* is partial, thus a ‘*the*’ is needed, similarly for *dhops* and *addpreRT*.
2. *precs* is partial.
3.  $\sigma_{p\text{-flag}}(\text{store}, \text{dip})$  is partial.
4. The routing table (*rt*) is modelled as a map ( $\text{ip} \Rightarrow r \text{ option}$ ) rather than a set of 7-tuples, likewise, the *r* is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address (*dip*) is taken from the argument to the function, rather than a part of the result. Well-definedness then follows from the structure of the type and more related facts are available automatically, rather than having to be acquired through tedious proofs.
5. Similar remarks hold for the *dests* mapping passed to *invalidate*, and *store*.

end

## 0.3 AODV protocol messages

```

theory Aodv_Message
  imports Aodv_Basic
  begin

```

```

datatype msg =
  Rreq nat rreqid ip sqn k ip sqn ip
  | Rrep nat ip sqn ip ip

```

```

/ Rerr "ip  $\rightarrow$  sqn" ip
/ Newpkt data ip
/ Pkt data ip ip

instantiation msg :: msg
begin
  definition newpkt_def [simp]: "newpkt  $\equiv$   $\lambda$ (d, dip). Newpkt d dip"
  definition eq_newpkt_def: "eq_newpkt m  $\equiv$  case m of Newpkt d dip  $\Rightarrow$  True | _  $\Rightarrow$  False"

  instance by intro_classes (simp add: eq_newpkt_def)
end

```

The *msg* type models the different messages used within AODV. The instantiation as a *msg* is a technicality due to the special treatment of *newpkt* messages in the AWN SOS rules. This use of classes allows a clean separation of the AWN-specific definitions and these AODV-specific definitions.

```

definition rreq :: "nat  $\times$  rreqid  $\times$  ip  $\times$  sqn  $\times$  k  $\times$  ip  $\times$  sqn  $\times$  ip  $\Rightarrow$  msg"
  where "rreq  $\equiv$   $\lambda$ (hops, rreqid, dip, dsn, dsk, oip, osn, sip).
    Rreq hops rreqid dip dsn dsk oip osn sip"

```

```

lemma rreq_simp [simp]:
  "rreq(hops, rreqid, dip, dsn, dsk, oip, osn, sip) = Rreq hops rreqid dip dsn dsk oip osn sip"
  unfolding rreq_def by simp

```

```

definition rrep :: "nat  $\times$  ip  $\times$  sqn  $\times$  ip  $\times$  ip  $\Rightarrow$  msg"
  where "rrep  $\equiv$   $\lambda$ (hops, dip, dsn, oip, sip). Rrep hops dip dsn oip sip"

```

```

lemma rrep_simp [simp]:
  "rrep(hops, dip, dsn, oip, sip) = Rrep hops dip dsn oip sip"
  unfolding rrep_def by simp

```

```

definition rerr :: "(ip  $\rightarrow$  sqn)  $\times$  ip  $\Rightarrow$  msg"
  where "rerr  $\equiv$   $\lambda$ (dests, sip). Rerr dests sip"

```

```

lemma rerr_simp [simp]:
  "rerr(dests, sip) = Rerr dests sip"
  unfolding rerr_def by simp

```

```

lemma not_eq_newpkt_rreq [simp]: " $\neg$ eq_newpkt (Rreq hops rreqid dip dsn dsk oip osn sip)"
  unfolding eq_newpkt_def by simp

```

```

lemma not_eq_newpkt_rrep [simp]: " $\neg$ eq_newpkt (Rrep hops dip dsn oip sip)"
  unfolding eq_newpkt_def by simp

```

```

lemma not_eq_newpkt_rerr [simp]: " $\neg$ eq_newpkt (Rerr dests sip)"
  unfolding eq_newpkt_def by simp

```

```

lemma not_eq_newpkt_pkt [simp]: " $\neg$ eq_newpkt (Pkt d dip sip)"
  unfolding eq_newpkt_def by simp

```

```

definition pkt :: "data  $\times$  ip  $\times$  ip  $\Rightarrow$  msg"
  where "pkt  $\equiv$   $\lambda$ (d, dip, sip). Pkt d dip sip"

```

```

lemma pkt_simp [simp]:
  "pkt(d, dip, sip) = Pkt d dip sip"
  unfolding pkt_def by simp

```

**end**

## 0.4 The AODV protocol

```

theory Aodv
imports Aodv_Data Aodv_Message
  "../AWN/AWN_SOS_Labels" "../AWN/AWN_Invariants"

```



begin

### 0.4.1 Data state

```
record state =
  ip      :: "ip"
  sn      :: "sqn"
  rt      :: "rt"
  rreqs  :: "(ip × rreqid) set"
  store  :: "store"

  msg     :: "msg"
  data    :: "data"
  dests   :: "ip → sqn"
  pre     :: "ip set"
  rreqid  :: "rreqid"
  dip     :: "ip"
  oip     :: "ip"
  hops    :: "nat"
  dsn     :: "sqn"
  dsk     :: "k"
  osn     :: "sqn"
  sip     :: "ip"
```

abbreviation aadv\_init :: "ip ⇒ state"

```
where "aadv_init i ≡ (|
  ip = i,
  sn = 1,
  rt = empty,
  rreqs = {},
  store = empty,

  msg   = (SOME x. True),
  data  = (SOME x. True),
  dests = (SOME x. True),
  pre   = (SOME x. True),
  rreqid = (SOME x. True),
  dip   = (SOME x. True),
  oip   = (SOME x. True),
  hops  = (SOME x. True),
  dsn   = (SOME x. True),
  dsk   = (SOME x. True),
  osn   = (SOME x. True),
  sip   = (SOME x. x ≠ i)
|)"
```

lemma some\_neq\_not\_eq [simp]: "¬((SOME x :: nat. x ≠ i) = i)"  
by (subst some\_eq\_ex) (metis zero\_neq\_numeral)

definition clear\_locals :: "state ⇒ state"

```
where "clear_locals ξ = ξ (|
  msg   := (SOME x. True),
  data  := (SOME x. True),
  dests := (SOME x. True),
  pre   := (SOME x. True),
  rreqid := (SOME x. True),
  dip   := (SOME x. True),
  oip   := (SOME x. True),
  hops  := (SOME x. True),
  dsn   := (SOME x. True),
  dsk   := (SOME x. True),
  osn   := (SOME x. True),
  sip   := (SOME x. x ≠ ip ξ)
|)"
```

```
lemma clear_locals_sip_not_ip [simp]: "¬(sip (clear_locals ξ) = ip ξ)"
  unfolding clear_locals_def by simp
```

```
lemma clear_locals_but_not_globals [simp]:
  "ip (clear_locals ξ) = ip ξ"
  "sn (clear_locals ξ) = sn ξ"
  "rt (clear_locals ξ) = rt ξ"
  "rreqs (clear_locals ξ) = rreqs ξ"
  "store (clear_locals ξ) = store ξ"
  unfolding clear_locals_def by auto
```

## 0.4.2 Auxilliary message handling definitions

```
definition is_newpkt
where "is_newpkt ξ ≡ case msg ξ of
      Newpkt data' dip' ⇒ { ξ(|data := data', dip := dip'|) }
      | _ ⇒ {}"
```

```
definition is_pkt
where "is_pkt ξ ≡ case msg ξ of
      Pkt data' dip' oip' ⇒ { ξ(|data := data', dip := dip', oip := oip'|) }
      | _ ⇒ {}"
```

```
definition is_rreq
where "is_rreq ξ ≡ case msg ξ of
      Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' ⇒
        { ξ(|hops := hops', rreqid := rreqid', dip := dip', dsn := dsn',
            dsk := dsk', oip := oip', osn := osn', sip := sip'|) }
      | _ ⇒ {}"
```

```
lemma is_rreq_asm [dest!]:
  assumes "ξ' ∈ is_rreq ξ"
  shows "(∃hops' rreqid' dip' dsn' dsk' oip' osn' sip'.
    msg ξ = Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' ∧
    ξ' = ξ(|hops := hops', rreqid := rreqid', dip := dip', dsn := dsn',
    dsk := dsk', oip := oip', osn := osn', sip := sip'|))"
  using assms unfolding is_rreq_def
  by (cases "msg ξ") simp_all
```

```
definition is_rrep
where "is_rrep ξ ≡ case msg ξ of
      Rrep hops' dip' dsn' oip' sip' ⇒
        { ξ(|hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip'|) }
      | _ ⇒ {}"
```

```
lemma is_rrep_asm [dest!]:
  assumes "ξ' ∈ is_rrep ξ"
  shows "(∃hops' dip' dsn' oip' sip'.
    msg ξ = Rrep hops' dip' dsn' oip' sip' ∧
    ξ' = ξ(|hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip'|))"
  using assms unfolding is_rrep_def
  by (cases "msg ξ") simp_all
```

```
definition is_rerr
where "is_rerr ξ ≡ case msg ξ of
      Rerr dests' sip' ⇒ { ξ(|dests := dests', sip := sip'|) }
      | _ ⇒ {}"
```

```
lemma is_rerr_asm [dest!]:
  assumes "ξ' ∈ is_rerr ξ"
  shows "(∃dests' sip'.
    msg ξ = Rerr dests' sip' ∧
    ξ' = ξ(|dests := dests', sip := sip'|))"
```

```

using assms unfolding is_rerr_def
by (cases "msg  $\xi$ ") simp_all

lemmas is_msg_defs =
  is_rerr_def is_rrep_def is_rreq_def is_pkt_def is_newpkt_def

lemma is_msg_inv_ip [simp]:
  " $\xi' \in \text{is\_rerr } \xi \implies \text{ip } \xi' = \text{ip } \xi$ "
  " $\xi' \in \text{is\_rrep } \xi \implies \text{ip } \xi' = \text{ip } \xi$ "
  " $\xi' \in \text{is\_rreq } \xi \implies \text{ip } \xi' = \text{ip } \xi$ "
  " $\xi' \in \text{is\_pkt } \xi \implies \text{ip } \xi' = \text{ip } \xi$ "
  " $\xi' \in \text{is\_newpkt } \xi \implies \text{ip } \xi' = \text{ip } \xi$ "
  unfolding is_msg_defs
  by (cases "msg  $\xi$ ", clarsimp+)+

lemma is_msg_inv_sn [simp]:
  " $\xi' \in \text{is\_rerr } \xi \implies \text{sn } \xi' = \text{sn } \xi$ "
  " $\xi' \in \text{is\_rrep } \xi \implies \text{sn } \xi' = \text{sn } \xi$ "
  " $\xi' \in \text{is\_rreq } \xi \implies \text{sn } \xi' = \text{sn } \xi$ "
  " $\xi' \in \text{is\_pkt } \xi \implies \text{sn } \xi' = \text{sn } \xi$ "
  " $\xi' \in \text{is\_newpkt } \xi \implies \text{sn } \xi' = \text{sn } \xi$ "
  unfolding is_msg_defs
  by (cases "msg  $\xi$ ", clarsimp+)+

lemma is_msg_inv_rt [simp]:
  " $\xi' \in \text{is\_rerr } \xi \implies \text{rt } \xi' = \text{rt } \xi$ "
  " $\xi' \in \text{is\_rrep } \xi \implies \text{rt } \xi' = \text{rt } \xi$ "
  " $\xi' \in \text{is\_rreq } \xi \implies \text{rt } \xi' = \text{rt } \xi$ "
  " $\xi' \in \text{is\_pkt } \xi \implies \text{rt } \xi' = \text{rt } \xi$ "
  " $\xi' \in \text{is\_newpkt } \xi \implies \text{rt } \xi' = \text{rt } \xi$ "
  unfolding is_msg_defs
  by (cases "msg  $\xi$ ", clarsimp+)+

lemma is_msg_inv_rreqs [simp]:
  " $\xi' \in \text{is\_rerr } \xi \implies \text{rreqs } \xi' = \text{rreqs } \xi$ "
  " $\xi' \in \text{is\_rrep } \xi \implies \text{rreqs } \xi' = \text{rreqs } \xi$ "
  " $\xi' \in \text{is\_rreq } \xi \implies \text{rreqs } \xi' = \text{rreqs } \xi$ "
  " $\xi' \in \text{is\_pkt } \xi \implies \text{rreqs } \xi' = \text{rreqs } \xi$ "
  " $\xi' \in \text{is\_newpkt } \xi \implies \text{rreqs } \xi' = \text{rreqs } \xi$ "
  unfolding is_msg_defs
  by (cases "msg  $\xi$ ", clarsimp+)+

lemma is_msg_inv_store [simp]:
  " $\xi' \in \text{is\_rerr } \xi \implies \text{store } \xi' = \text{store } \xi$ "
  " $\xi' \in \text{is\_rrep } \xi \implies \text{store } \xi' = \text{store } \xi$ "
  " $\xi' \in \text{is\_rreq } \xi \implies \text{store } \xi' = \text{store } \xi$ "
  " $\xi' \in \text{is\_pkt } \xi \implies \text{store } \xi' = \text{store } \xi$ "
  " $\xi' \in \text{is\_newpkt } \xi \implies \text{store } \xi' = \text{store } \xi$ "
  unfolding is_msg_defs
  by (cases "msg  $\xi$ ", clarsimp+)+

lemma is_msg_inv_sip [simp]:
  " $\xi' \in \text{is\_pkt } \xi \implies \text{sip } \xi' = \text{sip } \xi$ "
  " $\xi' \in \text{is\_newpkt } \xi \implies \text{sip } \xi' = \text{sip } \xi$ "
  unfolding is_msg_defs
  by (cases "msg  $\xi$ ", clarsimp+)+

```

### 0.4.3 The protocol process

```

datatype pseqp =
  PAadv
  | PNewPkt
  | PPkt
  | PRreq

```

```

| PRrep
| PRerr

fun nat_of_seqp :: "pseqp  $\Rightarrow$  nat"
where
  "nat_of_seqp PAodv = 1"
| "nat_of_seqp PPkt = 2"
| "nat_of_seqp PNewPkt = 3"
| "nat_of_seqp PRreq = 4"
| "nat_of_seqp PRrep = 5"
| "nat_of_seqp PRerr = 6"

instantiation "pseqp" :: ord
begin
definition less_eq_seqp [iff]: "l1  $\leq$  l2 = (nat_of_seqp l1  $\leq$  nat_of_seqp l2)"
definition less_seqp [iff]: "l1 < l2 = (nat_of_seqp l1 < nat_of_seqp l2)"
instance ..
end

abbreviation AODV
where
  "AODV  $\equiv$   $\lambda$ _. [[clear_locals]] call(PAodv)"

abbreviation PKT
where
  "PKT args  $\equiv$ 

  [[ $\xi$ . let (data, dip, oip) = args  $\xi$  in
    (clear_locals  $\xi$ ) ( $\lambda$  data := data, dip := dip, oip := oip  $\lambda$ )]
  call(PPkt)"]

abbreviation NEWPKT
where
  "NEWPKT args  $\equiv$ 

  [[ $\xi$ . let (data, dip) = args  $\xi$  in
    (clear_locals  $\xi$ ) ( $\lambda$  data := data, dip := dip  $\lambda$ )]
  call(PNewPkt)"]

abbreviation RREQ
where
  "RREQ args  $\equiv$ 

  [[ $\xi$ . let (hops, rreqid, dip, dsn, dsk, oip, osn, sip) = args  $\xi$  in
    (clear_locals  $\xi$ ) ( $\lambda$  hops := hops, rreqid := rreqid, dip := dip,
      dsn := dsn, dsk := dsk, oip := oip,
      osn := osn, sip := sip  $\lambda$ )]
  call(PRreq)"]

abbreviation RREP
where
  "RREP args  $\equiv$ 

  [[ $\xi$ . let (hops, dip, dsn, oip, sip) = args  $\xi$  in
    (clear_locals  $\xi$ ) ( $\lambda$  hops := hops, dip := dip, dsn := dsn,
      oip := oip, sip := sip  $\lambda$ )]
  call(PRrep)"]

abbreviation RERR
where
  "RERR args  $\equiv$ 

  [[ $\xi$ . let (dests, sip) = args  $\xi$  in
    (clear_locals  $\xi$ ) ( $\lambda$  dests := dests, sip := sip  $\lambda$ )]
  call(PRerr)"]

fun  $\Gamma_{AODV}$  :: "(state, msg, pseqp, pseqp label) seqp_env"
where
  " $\Gamma_{AODV}$  PAodv = labelled PAodv ("

```

```

receive(λmsg' ξ. ξ (| msg := msg' |)).
(
  <is_newpkt> NEWPKT(λξ. (data ξ, ip ξ))
  ⊕ <is_pkt> PKT(λξ. (data ξ, dip ξ, oip ξ))
  ⊕ <is_rreq>
    [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, { }) |)]
    RREQ(λξ. (hops ξ, rreqid ξ, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, sip ξ))
  ⊕ <is_rrep>
    [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, { }) |)]
    RREP(λξ. (hops ξ, dip ξ, dsn ξ, oip ξ, sip ξ))
  ⊕ <is_rerr>
    [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, { }) |)]
    RERR(λξ. (dests ξ, sip ξ))
)
⊕ <λξ. { ξ(| dip := dip |) | dip. dip ∈ qD(store ξ) ∩ vD(rt ξ) }>
  [[ξ. ξ (| data := hd(σqueue(store ξ, dip ξ)) |)]
  unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, ip ξ)).
  [[ξ. ξ (| store := the (drop (dip ξ) (store ξ)) |)]
  AODV()
  ▷ [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
    then Some (inc (sqn (rt ξ) rip)) else None) |)]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]
    [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } |)]
    [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
    then (dests ξ) rip else None) |)]
    groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)). AODV()
⊕ <λξ. { ξ(| dip := dip |)
  | dip. dip ∈ qD(store ξ) - vD(rt ξ) ∧ the (σp-flag(store ξ, dip)) = req }>
  [[ξ. ξ (| store := unsetRRF (store ξ) (dip ξ) |)]
  [[ξ. ξ (| sn := inc (sn ξ) |)]
  [[ξ. ξ (| rreqid := nrreqid (rreqs ξ) (ip ξ) |)]
  [[ξ. ξ (| rreqs := rreqs ξ ∪ {(ip ξ, rreqid ξ)} |)]
  broadcast(λξ. rreq(0, rreqid ξ, dip ξ, sqn (rt ξ) (dip ξ), sqnf (rt ξ) (dip ξ),
    ip ξ, sn ξ, ip ξ)). AODV()"

| "ΓAODV PNewPkt = labelled PNewPkt (
  <ξ. dip ξ = ip ξ>
  deliver(λξ. data ξ).AODV()
  ⊕ <ξ. dip ξ ≠ ip ξ>
  [[ξ. ξ (| store := add (data ξ) (dip ξ) (store ξ) |)]
  AODV()"

| "ΓAODV PPkt = labelled PPkt (
  <ξ. dip ξ = ip ξ>
  deliver(λξ. data ξ).AODV()
  ⊕ <ξ. dip ξ ≠ ip ξ>
  (
    <ξ. dip ξ ∈ vD (rt ξ)>
    unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, oip ξ)).AODV()
    ▷
    [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
    then Some (inc (sqn (rt ξ) rip)) else None) |)]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]
    [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } |)]
    [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
    then (dests ξ) rip else None) |)]
    groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
  ⊕ <ξ. dip ξ ∉ vD (rt ξ)>
  (
    <ξ. dip ξ ∈ iD (rt ξ)>
    groupcast(λξ. the (precs (rt ξ) (dip ξ)),
      λξ. rerr([dip ξ ↦ sqn (rt ξ) (dip ξ)], ip ξ)). AODV()
    ⊕ <ξ. dip ξ ∉ iD (rt ξ)>

```

```

    AODV()
  )
))"

| "ΓAODV PRreq = labelled PRreq (
  ⟨ξ. (oip ξ, rreqid ξ) ∈ rreqs ξ⟩
  AODV()
  ⊕ ⟨ξ. (oip ξ, rreqid ξ) ∉ rreqs ξ⟩
  [[ξ. ξ (| rt := update (rt ξ) (oip ξ) (osn ξ, kno, val, hops ξ + 1, sip ξ, { }) )]]
  [[ξ. ξ (| rreqs := rreqs ξ ∪ {(oip ξ, rreqid ξ)} )]]
  (
    ⟨ξ. dip ξ = ip ξ⟩
    [[ξ. ξ (| sn := max (sn ξ) (dsn ξ) )]]
    unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(0, dip ξ, sn ξ, oip ξ, ip ξ)).AODV()
  ▷
    [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
      then Some (inc (sqn (rt ξ) rip)) else None) )]]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
    [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
    [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
      then (dests ξ) rip else None) )]]
    groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
  ⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
  (
    ⟨ξ. dip ξ ∈ vD (rt ξ) ∧ dsn ξ ≤ sqn (rt ξ) (dip ξ) ∧ sqnf (rt ξ) (dip ξ) = kno⟩
    [[ξ. ξ (| rt := the (addpreRT (rt ξ) (dip ξ) {sip ξ}) )]]
    [[ξ. ξ (| rt := the (addpreRT (rt ξ) (oip ξ) {the (nhop (rt ξ) (dip ξ))}) )]]
    unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(the (dhops (rt ξ) (dip ξ)), dip ξ,
      sqn (rt ξ) (dip ξ), oip ξ, ip ξ)).
    AODV()
  ▷
    [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
      then Some (inc (sqn (rt ξ) rip)) else None) )]]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
    [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
    [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
      then (dests ξ) rip else None) )]]
    groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
  ⊕ ⟨ξ. dip ξ ∉ vD (rt ξ) ∨ sqn (rt ξ) (dip ξ) < dsn ξ ∨ sqnf (rt ξ) (dip ξ) = unk⟩
    broadcast(λξ. rreq(hops ξ + 1, rreqid ξ, dip ξ, max (sqn (rt ξ) (dip ξ)) (dsn ξ),
      dsk ξ, oip ξ, osn ξ, ip ξ)).
    AODV()
  )
))"

| "ΓAODV PRrep = labelled PRrep (
  ⟨ξ. rt ξ ≠ update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, { }) ⟩
  (
    [[ξ. ξ (| rt := update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, { }) )]]
  (
    ⟨ξ. oip ξ = ip ξ ⟩
    AODV()
  ⊕ ⟨ξ. oip ξ ≠ ip ξ ⟩
  (
    ⟨ξ. oip ξ ∈ vD (rt ξ) ⟩
    [[ξ. ξ (| rt := the (addpreRT (rt ξ) (dip ξ) {the (nhop (rt ξ) (oip ξ))}) )]]
    [[ξ. ξ (| rt := the (addpreRT (rt ξ) (the (nhop (rt ξ) (dip ξ)))
      {the (nhop (rt ξ) (oip ξ))}) )]]
    unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(hops ξ + 1, dip ξ, dsn ξ, oip ξ, ip ξ)).
    AODV()
  ▷
    [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))

```

```

        then Some (inc (sqn (rt ξ) rip)) else None) ]]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]
    [[ξ. ξ (| pre := ⋃{ the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } |)]
    [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ {})
        then (dests ξ) rip else None) |)]
    groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
  ⊕ ⟨ξ. oip ξ ∉ vD (rt ξ)⟩
    AODV()
)
)
)
⊕ ⟨ξ. rt ξ = update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {})⟩
  AODV()
)"

```

```

| "ΓAODV PRerr = labelled PRerr (
  [[ξ. ξ (| dests := (λrip. case (dests ξ) rip of None ⇒ None
    | Some rsn ⇒ if rip ∈ vD (rt ξ) ∧ the (nhop (rt ξ) rip) = sip ξ
      ∧ sqn (rt ξ) rip < rsn then Some rsn else None) |)]
  [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]
  [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]
  [[ξ. ξ (| pre := ⋃{ the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } |)]
  [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ {})
    then (dests ξ) rip else None) |)]
  groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)). AODV()"

```

```

declare ΓAODV.simps [simp del, code del]
lemmas ΓAODV.simps [simp, code] = ΓAODV.simps [simplified]

```

```

fun ΓAODV.skeleton
where
  "ΓAODV.skeleton PAadv = seqp_skeleton (ΓAODV PAadv)"
  | "ΓAODV.skeleton PNewPkt = seqp_skeleton (ΓAODV PNewPkt)"
  | "ΓAODV.skeleton PPkt = seqp_skeleton (ΓAODV PPkt)"
  | "ΓAODV.skeleton PRreq = seqp_skeleton (ΓAODV PRreq)"
  | "ΓAODV.skeleton PRrep = seqp_skeleton (ΓAODV PRrep)"
  | "ΓAODV.skeleton PRerr = seqp_skeleton (ΓAODV PRerr)"

```

```

lemma ΓAODV.skeleton_wf [simp]:
  "wellformed ΓAODV.skeleton"
proof (rule, intro allI)
  fix pn pn'
  show "call(pn') ∉ stermsl (ΓAODV.skeleton pn)"
  by (cases pn) simp_all
qed

```

```

declare ΓAODV.skeleton.simps [simp del, code del]
lemmas ΓAODV.skeleton.simps [simp, code]
  = ΓAODV.skeleton.simps [simplified ΓAODV.simps seqp_skeleton.simps]

```

```

lemma aodv_proc_cases [dest]:
  fixes p pn
  shows "p ∈ ctermsl (ΓAODV pn) ⇒
    (p ∈ ctermsl (ΓAODV PAadv) ∨
     p ∈ ctermsl (ΓAODV PNewPkt) ∨
     p ∈ ctermsl (ΓAODV PPkt) ∨
     p ∈ ctermsl (ΓAODV PRreq) ∨
     p ∈ ctermsl (ΓAODV PRrep) ∨
     p ∈ ctermsl (ΓAODV PRerr))"
  using assms
  by (cases pn) simp_all

```

```

definition σAODV :: "ip ⇒ (state × (state, msg, pseqp, pseqp label) seqp) set"

```

where  $\sigma_{AODV} i \equiv \{(aadv\_init\ i, \Gamma_{AODV} PAadv)\}$ "

abbreviation *paadv*

$:: ip \Rightarrow (state \times (state, msg, pseqp, pseqp\ label)\ seqp, msg\ seq\_action)\ automaton$ "

where

"*paadv*  $i \equiv (\mid\ init = \sigma_{AODV} i, trans = seqp\_sos\ \Gamma_{AODV}\ \mid)$ "

lemma *aadv\_trans*: "*trans* (*paadv*  $i$ ) = *seqp\\_sos*  $\Gamma_{AODV}$ "

by *simp*

lemma *aadv\_control\_within* [*simp*]: "*control\_within*  $\Gamma_{AODV}$  (*init* (*paadv*  $i$ ))"

*unfolding*  $\sigma_{AODV\_def}$  by (*rule* *control\_withinI*) (*auto* *simp* *del*:  $\Gamma_{AODV\_simps}$ )

lemma *aadv\_wf* [*simp*]:

"*wellformed*  $\Gamma_{AODV}$ "

*proof* (*rule*, *intro* *allI*)

fix  $pn\ pn'$

show "*call*( $pn'$ )  $\notin\ sterms1$  ( $\Gamma_{AODV}\ pn$ )"

by (*cases*  $pn$ ) *simp\_all*

qed

lemmas *aadv\_labels\_not\_empty* [*simp*] = *labels\_not\_empty* [*OF* *aadv\_wf*]

lemma *aadv\_ex\_label* [*intro*]: " $\exists l. l \in labels\ \Gamma_{AODV}\ p$ "

by (*metis* *aadv\_labels\_not\_empty* *all\_not\_in\_conv*)

lemma *aadv\_ex\_labelE* [*elim*]:

assumes " $\forall l \in labels\ \Gamma_{AODV}\ p. P\ l\ p$ "

and " $\exists p\ l. P\ l\ p \implies Q$ "

shows " $Q$ "

using *assms* by (*metis* *aadv\_ex\_label*)

lemma *aadv\_simple\_labels* [*simp*]: "*simple\_labels*  $\Gamma_{AODV}$ "

*proof*

fix  $pn\ p$

assume " $p \in subterms(\Gamma_{AODV}\ pn)$ "

thus " $\exists ! l. labels\ \Gamma_{AODV}\ p = \{l\}$ "

by (*cases*  $pn$ ) (*simp\_all* *cong*: *seqp\_congs* | *elim* *disjE*)+

qed

lemma  $\sigma_{AODV\_labels}$  [*simp*]: " $(\xi, p) \in \sigma_{AODV}\ i \implies labels\ \Gamma_{AODV}\ p = \{PAadv-:0\}$ "

*unfolding*  $\sigma_{AODV\_def}$  by *simp*

lemma *aadv\_init\_kD\_empty* [*simp*]:

" $(\xi, p) \in \sigma_{AODV}\ i \implies kD\ (rt\ \xi) = \{\}$ "

*unfolding*  $\sigma_{AODV\_def}$   $kD\_def$  by *simp*

lemma *aadv\_init\_sip\_not\_ip* [*simp*]: " $\neg(sip\ (aadv\_init\ i) = i)$ " by *simp*

lemma *aadv\_init\_sip\_not\_ip'* [*simp*]:

assumes " $(\xi, p) \in \sigma_{AODV}\ i$ "

shows " $sip\ \xi \neq ip\ \xi$ "

using *assms* *unfolding*  $\sigma_{AODV\_def}$  by *simp*

lemma *aadv\_init\_sip\_not\_i* [*simp*]:

assumes " $(\xi, p) \in \sigma_{AODV}\ i$ "

shows " $sip\ \xi \neq i$ "

using *assms* *unfolding*  $\sigma_{AODV\_def}$  by *simp*

lemma *clear\_locals\_sip\_not\_ip'*:

assumes " $ip\ \xi = i$ "

shows " $\neg(sip\ (clear\_locals\ \xi) = i)$ "

using *assms* by *auto*



Stop the simplifier from descending into process terms.

```
declare seqp_congs [cong]
```

Configure the main invariant tactic for AODV.

```
declare
```

```
  ΓAODV_simps [ctermenv]
  aodv_proc_cases [ctermenv_cases]
  seq_invariant_ctermenvI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans,
                           ctermenv_intros]
  seq_step_invariant_ctermenvI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans,
                                ctermenv_intros]
```

```
end
```

## 0.5 Invariant assumptions and properties

```
theory Aodv_Predicates
```

```
imports Aodv
```

```
begin
```

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

```
abbreviation not_Pkt :: "msg ⇒ bool"
```

```
where "not_Pkt m ≡ case m of Pkt _ _ _ ⇒ False | _ ⇒ True"
```

```
definition msg_sender :: "msg ⇒ ipc"
```

```
where "msg_sender m ≡ case m of Rreq _ _ _ _ _ ipc ⇒ ipc
    | Rrep _ _ _ _ ipc ⇒ ipc
    | Rerr _ ipc ⇒ ipc
    | Pkt _ _ ipc ⇒ ipc"
```

```
lemma msg_sender_simps [simp]:
```

```
"∧hops rreqid dip dsn dsk oip osn sip.
  msg_sender (Rreq hops rreqid dip dsn dsk oip osn sip) = sip"
"∧hops dip dsn oip sip. msg_sender (Rrep hops dip dsn oip sip) = sip"
"∧dests sip. msg_sender (Rerr dests sip) = sip"
"∧dip sip. msg_sender (Pkt d dip sip) = sip"
unfolding msg_sender_def by simp_all
```

```
definition msg_zhops :: "msg ⇒ bool"
```

```
where "msg_zhops m ≡ case m of
    Rreq hopsc _ dipc _ _ oipcc _ sipc ⇒ hopsc = 0 → oipcc = sipc
  | Rrep hopsc dipc _ _ sipc ⇒ hopsc = 0 → dipc = sipc
  | _ ⇒ True"
```

```
lemma msg_zhops_simps [simp]:
```

```
"∧hops rreqid dip dsn dsk oip osn sip.
  msg_zhops (Rreq hops rreqid dip dsn dsk oip osn sip) = (hops = 0 → oip = sip)"
"∧hops dip dsn oip sip. msg_zhops (Rrep hops dip dsn oip sip) = (hops = 0 → dip = sip)"
"∧dests sip. msg_zhops (Rerr dests sip) = True"
"∧dip. msg_zhops (Newpkt d dip) = True"
"∧dip sip. msg_zhops (Pkt d dip sip) = True"
unfolding msg_zhops_def by simp_all
```

```
definition rreq_rrep_sn :: "msg ⇒ bool"
```

```
where "rreq_rrep_sn m ≡ case m of Rreq _ _ _ _ _ osnc _ ⇒ osnc ≥ 1
    | Rrep _ _ dsnc _ _ ⇒ dsnc ≥ 1
    | _ ⇒ True"
```

```
lemma rreq_rrep_sn_simps [simp]:
```

```
"∧hops rreqid dip dsn dsk oip osn sip.
  rreq_rrep_sn (Rreq hops rreqid dip dsn dsk oip osn sip) = (osn ≥ 1)"
"∧hops dip dsn oip sip. rreq_rrep_sn (Rrep hops dip dsn oip sip) = (dsn ≥ 1)"
"∧dests sip. rreq_rrep_sn (Rerr dests sip) = True"
```

```

"∧d dip.          rreq_rrep_sn (Newpkt d dip) = True"
"∧d dip sip.     rreq_rrep_sn (Pkt d dip sip) = True"
unfolding rreq_rrep_sn_def by simp_all

```

**definition** `rreq_rrep_fresh` :: "rt ⇒ msg ⇒ bool"

```

where "rreq_rrep_fresh crt m ≡ case m of Rreq hopsc _ _ _ oipc osnc ipcc ⇒ (ipcc ≠ oipc →
    oipc ∈ kD(crt) ∧ (sqn crt oipc > osnc
        ∨ (sqn crt oipc = osnc
            ∧ the (dhops crt oipc) ≤ hopsc
            ∧ the (flag crt oipc) = val)))
    | Rrep hopsc dipc dsnc _ ipcc ⇒ (ipcc ≠ dipc →
        dipc ∈ kD(crt)
        ∧ sqn crt dipc = dsnc
        ∧ the (dhops crt dipc) = hopsc
        ∧ the (flag crt dipc) = val)
    | _ ⇒ True"

```

**lemma** `rreq_rrep_fresh [simp]`:

```

"∧hops rreqid dip dsn dsk oip osn sip.
  rreq_rrep_fresh crt (Rreq hops rreqid dip dsn dsk oip osn sip) =
  (sip ≠ oip → oip ∈ kD(crt)
    ∧ (sqn crt oip > osn
      ∨ (sqn crt oip = osn
        ∧ the (dhops crt oip) ≤ hops
        ∧ the (flag crt oip) = val)))"
"∧hops dip dsn oip sip. rreq_rrep_fresh crt (Rrep hops dip dsn oip sip) =
  (sip ≠ dip → dip ∈ kD(crt)
    ∧ sqn crt dip = dsn
    ∧ the (dhops crt dip) = hops
    ∧ the (flag crt dip) = val)"
"∧dests sip.          rreq_rrep_fresh crt (Rerr dests sip) = True"
"∧d dip.             rreq_rrep_fresh crt (Newpkt d dip) = True"
"∧d dip sip.        rreq_rrep_fresh crt (Pkt d dip sip) = True"
unfolding rreq_rrep_fresh_def by simp_all

```

**definition** `rerr_invalid` :: "rt ⇒ msg ⇒ bool"

```

where "rerr_invalid crt m ≡ case m of Rerr destsc _ ⇒ (∀ripc ∈ dom(destsc).
    (ripc ∈ iD(crt) ∧ the (destsc ripc) = sqn crt ripc)
    | _ ⇒ True"

```

**lemma** `rerr_invalid [simp]`:

```

"∧hops rreqid dip dsn dsk oip osn sip.
  rerr_invalid crt (Rreq hops rreqid dip dsn dsk oip osn sip) = True"
"∧hops dip dsn oip sip. rerr_invalid crt (Rrep hops dip dsn oip sip) = True"
"∧dests sip.          rerr_invalid crt (Rerr dests sip) = (∀ripc ∈ dom(dests).
    ripc ∈ iD(crt) ∧ the (dests ripc) = sqn crt ripc)"
"∧d dip.             rerr_invalid crt (Newpkt d dip) = True"
"∧d dip sip.        rerr_invalid crt (Pkt d dip sip) = True"
unfolding rerr_invalid_def by simp_all

```

**definition**

```

initmissing :: "(nat ⇒ state option) × 'a ⇒ (nat ⇒ state) × 'a"

```

where

```

"initmissing σ = (λi. case (fst σ) i of None ⇒ aadv_init i | Some s ⇒ s, snd σ)"

```

**lemma** `not_in_net_ips_fst_init_missing [simp]`:

```

assumes "i ∉ net_ips σ"
  shows "fst (initmissing (netgmap fst σ)) i = aadv_init i"
using assms unfolding initmissing_def by simp

```

**lemma** `fst_initmissing_netgmap_pair_fst [simp]`:

```

"fst (initmissing (netgmap (λ(p, q). (fst (id p), snd (id p), q)) s))
  = fst (initmissing (netgmap fst s))"

```

unfolding initmissing\_def by auto

We introduce a streamlined alternative to `initmissing` with `netgmap` to simplify invariant statements and thus facilitate their comprehension and presentation.

```
lemma fst_initmissing_netgmap_default_aodv_init_netlift:
  "fst (initmissing (netgmap fst s)) = default aodv_init (netlift fst s)"
  unfolding initmissing_def default_def
  by (simp add: fst_netgmap_netlift del: One_nat_def)
```

**definition**

```
netglobal :: "(nat  $\Rightarrow$  state)  $\Rightarrow$  bool  $\Rightarrow$  ((state  $\times$  'b)  $\times$  'c) net_state  $\Rightarrow$  bool"
where
  "netglobal P  $\equiv$  ( $\lambda$ s. P (default aodv_init (netlift fst s)))"
```

end

## 0.6 Quality relations between routes

```
theory Fresher
imports Aodv_Data
begin
```

### 0.6.1 Net sequence numbers

**On individual routes**

**definition**

```
nsqn_r :: "r  $\Rightarrow$  sqn"
where
  "nsqn_r r  $\equiv$  if  $\pi_4(r) = \text{val} \vee \pi_2(r) = 0$  then  $\pi_2(r)$  else ( $\pi_2(r) - 1$ )"
```

**lemma nsqn\_r\_def'**:

```
"nsqn_r r = (if  $\pi_4(r) = \text{inv}$  then  $\pi_2(r) - 1$  else  $\pi_2(r)$ )"
unfolding nsqn_r_def by simp
```

**lemma nsqn\_r\_zero [simp]**:

```
" $\bigwedge$ dsn dsk flag hops nhip pre. nsqn_r (0, dsk, flag, hops, nhip, pre) = 0"
unfolding nsqn_r_def by clarsimp
```

**lemma nsqn\_r\_val [simp]**:

```
" $\bigwedge$ dsn dsk hops nhip pre. nsqn_r (dsn, dsk, val, hops, nhip, pre) = dsn"
unfolding nsqn_r_def by clarsimp
```

**lemma nsqn\_r\_inv [simp]**:

```
" $\bigwedge$ dsn dsk hops nhip pre. nsqn_r (dsn, dsk, inv, hops, nhip, pre) = dsn - 1"
unfolding nsqn_r_def by clarsimp
```

**lemma nsqn\_r\_lte\_dsn [simp]**:

```
" $\bigwedge$ dsn dsk flag hops nhip pre. nsqn_r (dsn, dsk, flag, hops, nhip, pre)  $\leq$  dsn"
unfolding nsqn_r_def by clarsimp
```

**On routes in routing tables**

**definition**

```
nsqn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  sqn"
where
  "nsqn  $\equiv$   $\lambda$ rt dip. case  $\sigma_{\text{route}}(rt, dip)$  of None  $\Rightarrow$  0 | Some r  $\Rightarrow$  nsqn_r(r)"
```

**lemma nsqn\_sqn\_def**:

```
" $\bigwedge$ rt dip. nsqn rt dip = (if flag rt dip = Some val  $\vee$  sqn rt dip = 0
  then sqn rt dip else sqn rt dip - 1)"
unfolding nsqn_def sqn_def by (clarsimp split: option.split) auto
```

**lemma not\_in\_kD\_nsqn [simp]**:

```
assumes "dip  $\notin$  kD(rt)"
shows "nsqn rt dip = 0"
```

```

using assms unfolding nsqn_def by simp

lemma kD_nsqn:
  assumes "dip ∈ kD(rt)"
  shows "nsqn rt dip = nsqn_r (the (σ_route(rt, dip)))"
  using assms [THEN kD_Some] unfolding nsqn_def by clarsimp

lemma nsqn_r_flag_pred [simp, intro]:
  fixes dsn dsk flag hops nhip pre
  assumes "P (nsqn_r (dsn, dsk, val, hops, nhip, pre))"
  and "P (nsqn_r (dsn, dsk, inv, hops, nhip, pre))"
  shows "P (nsqn_r (dsn, dsk, flag, hops, nhip, pre))"
  using assms by (cases flag) auto

lemma nsqn_r_addpreRT_inv [simp]:
  "∧rt dip npre dip'. dip ∈ kD(rt) ⇒
  nsqn_r (the (the (addpreRT rt dip npre) dip')) = nsqn_r (the (rt dip'))"
  unfolding addpreRT_def nsqn_r_def
  by (frule kD_Some) (clarsimp split: option.split)

lemma sqn_nsqn:
  "∧rt dip. sqn rt dip - 1 ≤ nsqn rt dip"
  unfolding sqn_def nsqn_def by (clarsimp split: option.split)

lemma nsqn_sqn: "nsqn rt dip ≤ sqn rt dip"
  unfolding sqn_def nsqn_def by (cases "rt dip") auto

lemma val_nsqn_sqn [elim, simp]:
  assumes "ip ∈ kD(rt)"
  and "the (flag rt ip) = val"
  shows "nsqn rt ip = sqn rt ip"
  using assms unfolding nsqn_sqn_def by auto

lemma vD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ vD(rt)"
  shows "nsqn rt ip = sqn rt ip"
  proof -
    from ⟨ip ∈ vD(rt)⟩ have "ip ∈ kD(rt)"
      and "the (flag rt ip) = val" by auto
    thus ?thesis ..
  qed

lemma inv_nsqn_sqn [elim, simp]:
  assumes "ip ∈ kD(rt)"
  and "the (flag rt ip) = inv"
  shows "nsqn rt ip = sqn rt ip - 1"
  using assms unfolding nsqn_sqn_def by auto

lemma iD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ iD(rt)"
  shows "nsqn rt ip = sqn rt ip - 1"
  proof -
    from ⟨ip ∈ iD(rt)⟩ have "ip ∈ kD(rt)"
      and "the (flag rt ip) = inv" by auto
    thus ?thesis ..
  qed

lemma nsqn_update_changed_kno_val [simp]: "∧rt ip dsn dsk hops nhip.
  rt ≠ update rt ip (dsn, kno, val, hops, nhip, {})
  ⇒ nsqn (update rt ip (dsn, kno, val, hops, nhip, {})) ip = dsn"
  using assms unfolding nsqn_r_def update_def
  by (clarsimp simp: kD_nsqn split: option.split_asm option.split split_if_asm)
  (metis fun_upd_triv)

```

```

lemma nsqn_addpreRT_inv [simp]:
  " $\bigwedge$ rt dip npre dip'. dip  $\in$  kD(rt)  $\implies$ 
  nsqn (the (addpreRT rt dip npre)) dip' = nsqn rt dip'"
  unfolding addpreRT_def nsqn_def nsqn_r_def
  by (frule kD_Some) (clarsimp split: option.split)

lemma nsqn_update_other [simp]:
  fixes dsn dsk flag hops dip nhip pre rt ip
  assumes "dip  $\neq$  ip"
  shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip, pre)) dip = nsqn rt dip"
  using assms unfolding nsqn_def
  by (clarsimp split: option.split)

lemma nsqn_invalidate_eq:
  assumes "dip  $\in$  kD(rt)"
    and "dests dip = Some rsn"
  shows "nsqn (invalidate rt dests) dip = rsn - 1"
  using assms
  proof -
    from assms obtain dsk hops nhip pre
      where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip, pre)"
    unfolding invalidate_def by auto
    moreover from (dip  $\in$  kD(rt)) have "dip  $\in$  kD(invalidate rt dests)" by simp
    ultimately show ?thesis
      using (dests dip = Some rsn) by simp
  qed

lemma nsqn_invalidate_other [simp]:
  assumes "dip  $\in$  kD(rt)"
    and "dip  $\notin$  dom dests"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
  using assms by (clarsimp simp add: kD_nsqn)

```

## 0.6.2 Comparing routes

definition

```
fresher :: "r  $\Rightarrow$  r  $\Rightarrow$  bool" ("(_/  $\sqsubseteq$  _)" [51, 51] 50)
```

where

```
"fresher r r'  $\equiv$  ((nsqn_r r < nsqn_r r')  $\vee$  (nsqn_r r = nsqn_r r'  $\wedge$   $\pi_5$ (r)  $\geq$   $\pi_5$ (r')))"
```

lemma fresherI1 [intro]:

```
assumes "nsqn_r r < nsqn_r r'"
  shows "r  $\sqsubseteq$  r'"
  unfolding fresher_def using assms by simp
```

lemma fresherI2 [intro]:

```
assumes "nsqn_r r = nsqn_r r'"
  and " $\pi_5$ (r)  $\geq$   $\pi_5$ (r'"
  shows "r  $\sqsubseteq$  r'"
  unfolding fresher_def using assms by simp
```

lemma fresherI [intro]:

```
assumes "(nsqn_r r < nsqn_r r')  $\vee$  (nsqn_r r = nsqn_r r'  $\wedge$   $\pi_5$ (r)  $\geq$   $\pi_5$ (r'))"
  shows "r  $\sqsubseteq$  r'"
  unfolding fresher_def using assms .
```

lemma fresherE [elim]:

```
assumes "r  $\sqsubseteq$  r'"
  and "nsqn_r r < nsqn_r r'  $\implies$  P r r'"
  and "nsqn_r r = nsqn_r r'  $\wedge$   $\pi_5$ (r)  $\geq$   $\pi_5$ (r')  $\implies$  P r r'"
  shows "P r r'"
  using assms unfolding fresher_def by auto
```

lemma fresher\_refl [simp]: "r  $\sqsubseteq$  r"

unfolding fresher\_def by simp

lemma fresher\_trans [elim, trans]:

"[[ x  $\sqsubseteq$  y; y  $\sqsubseteq$  z ]]  $\implies$  x  $\sqsubseteq$  z"  
unfolding fresher\_def by auto

lemma not\_fresher\_trans [elim, trans]:

"[[  $\neg$ (x  $\sqsubseteq$  y);  $\neg$ (z  $\sqsubseteq$  x) ]]  $\implies$   $\neg$ (z  $\sqsubseteq$  y)"  
unfolding fresher\_def by auto

lemma fresher\_dsn\_flag\_hops\_const [simp]:

fixes dsn dsk dsk' flag hops nhip nhip' pre pre'  
shows "(dsn, dsk, flag, hops, nhip, pre)  $\sqsubseteq$  (dsn, dsk', flag, hops, nhip', pre)"  
unfolding fresher\_def by (cases flag) simp\_all

lemma addpre\_fresher [simp]: " $\bigwedge$ r npre. r  $\sqsubseteq$  (addpre r npre)"

by clarsimp

### 0.6.3 Comparing routing tables

definition

rt\_fresher :: "ip  $\Rightarrow$  rt  $\Rightarrow$  rt  $\Rightarrow$  bool"

where

"rt\_fresher  $\equiv$   $\lambda$ dip rt rt'. (the ( $\sigma_{route}(rt, dip)$ ))  $\sqsubseteq$  (the ( $\sigma_{route}(rt', dip)$ ))"

abbreviation

rt\_fresher\_syn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  rt  $\Rightarrow$  bool" (" $\_ / \sqsubseteq \_$ " [51, 999, 51] 50)

where

"rt1  $\sqsubseteq_i$  rt2  $\equiv$  rt\_fresher i rt1 rt2"

lemma rt\_fresher\_def':

"(rt1  $\sqsubseteq_i$  rt2) = (nsqn<sub>r</sub> (the (rt1 i)) < nsqn<sub>r</sub> (the (rt2 i))  $\vee$   
nsqn<sub>r</sub> (the (rt1 i)) = nsqn<sub>r</sub> (the (rt2 i))  $\wedge$   $\pi_5$  (the (rt2 i))  $\leq$   $\pi_5$  (the (rt1 i)))"  
unfolding rt\_fresher\_def fresher\_def by (rule refl)

lemma single\_rt\_fresher [intro]:

assumes "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"  
shows "rt1  $\sqsubseteq_{ip}$  rt2"  
using assms unfolding rt\_fresher\_def .

lemma rt\_fresher\_single [intro]:

assumes "rt1  $\sqsubseteq_{ip}$  rt2"  
shows "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"  
using assms unfolding rt\_fresher\_def .

lemma rt\_fresher\_def2:

assumes "dip  $\in$  kD(rt1)"  
and "dip  $\in$  kD(rt2)"  
shows "(rt1  $\sqsubseteq_{dip}$  rt2) = (nsqn rt1 dip < nsqn rt2 dip  
 $\vee$  (nsqn rt1 dip = nsqn rt2 dip  
 $\wedge$  the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip)))"  
using assms unfolding rt\_fresher\_def fresher\_def by (simp add: kD\_nsqn proj5\_eq\_dhops)

lemma rt\_fresherI1 [intro]:

assumes "dip  $\in$  kD(rt1)"  
and "dip  $\in$  kD(rt2)"  
and "nsqn rt1 dip < nsqn rt2 dip"  
shows "rt1  $\sqsubseteq_{dip}$  rt2"  
unfolding rt\_fresher\_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt\_fresherI2 [intro]:

assumes "dip  $\in$  kD(rt1)"  
and "dip  $\in$  kD(rt2)"  
and "nsqn rt1 dip = nsqn rt2 dip"

```

    and "the (dhops rt1 dip) ≥ the (dhops rt2 dip)"
    shows "rt1  $\sqsubseteq_{dip}$  rt2"
  unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3-4) by simp

lemma rt_fresherE [elim]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "dip ∈ kD(rt1)"
    and "dip ∈ kD(rt2)"
    and "[[ nsqn rt1 dip < nsqn rt2 dip ] ] ⇒ P rt1 rt2 dip"
    and "[[ nsqn rt1 dip = nsqn rt2 dip;
      the (dhops rt1 dip) ≥ the (dhops rt2 dip) ] ] ⇒ P rt1 rt2 dip"
  shows "P rt1 rt2 dip"
  using assms(1) unfolding rt_fresher_def2 [OF assms(2-3)]
  using assms(4-5) by auto

lemma rt_fresher_refl [simp]: "rt  $\sqsubseteq_{dip}$  rt"
  unfolding rt_fresher_def by simp

lemma rt_fresher_trans [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
  using assms unfolding rt_fresher_def by auto

lemma rt_fresher_if_Some [intro!]:
  assumes "the (rt dip)  $\sqsubseteq$  r"
  shows "rt  $\sqsubseteq_{dip}$  (λip. if ip = dip then Some r else rt ip)"
  using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip ⇒ rt ⇒ rt ⇒ bool"
where
  "rt_fresh_as ≡ λdip rt1 rt2. (rt1  $\sqsubseteq_{dip}$  rt2) ∧ (rt2  $\sqsubseteq_{dip}$  rt1)"

abbreviation
  rt_fresh_as_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ≈_ _)" [51, 999, 51] 50)
where
  "rt1 ≈i rt2 ≡ rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_refl [simp]: "∧rt dip. rt ≈dip rt"
  unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
  "∧rt1 rt2 rt3 dip. [[ rt1 ≈dip rt2; rt2 ≈dip rt3 ] ] ⇒ rt1 ≈dip rt3"
  unfolding rt_fresh_as_def rt_fresher_def
  by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt1"
  shows "rt1 ≈dip rt2"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
  assumes "dip ∈ kD(rt1)"
    and "dip ∈ kD(rt2)"
    and "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"
    and "the (rt2 dip)  $\sqsubseteq$  the (rt1 dip)"
  shows "rt1 ≈dip rt2"
  using assms unfolding rt_fresh_as_def
  by (clarsimp dest!: single_rt_fresher)

lemma nsqn_rt_fresh_asI:
  assumes "dip ∈ kD(rt)"
    and "dip ∈ kD(rt′)"

```

```

    and "nsqn rt dip = nsqn rt' dip"
    and " $\pi_5(\text{the } (rt \text{ dip})) = \pi_5(\text{the } (rt' \text{ dip}))$ "
  shows "rt  $\approx_{dip}$  rt'"
proof
  from assms(1-2,4) have dhops': "the (dhops rt' dip)  $\leq$  the (dhops rt dip)"
  by (simp add: proj5_eq_dhops)
  with assms(1-3) show "rt  $\sqsubseteq_{dip}$  rt'"
  by (rule rt_fresherI2)
next
  from assms(1-2,4) have dhops: "the (dhops rt dip)  $\leq$  the (dhops rt' dip)"
  by (simp add: proj5_eq_dhops)
  with assms(2,1) assms(3) [symmetric] show "rt'  $\sqsubseteq_{dip}$  rt"
  by (rule rt_fresherI2)
qed

lemma rt_fresh_asE [elim]:
  assumes "rt1  $\approx_{dip}$  rt2"
    and "[[ rt1  $\sqsubseteq_{dip}$  rt2; rt2  $\sqsubseteq_{dip}$  rt1 ]  $\implies$  P rt1 rt2 dip]"
  shows "P rt1 rt2 dip"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD1 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\sqsubseteq_{dip}$  rt1"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\approx_{dip}$  rt1"
  using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
  assumes " $\neg$  (rt1  $\sqsubseteq_{dip}$  rt2)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
proof
  assume "rt1  $\approx_{dip}$  rt2"
  hence "rt1  $\sqsubseteq_{dip}$  rt2" ..
  with  $\langle \neg$  (rt1  $\sqsubseteq_{dip}$  rt2)  $\rangle$  show False ..
qed

lemma not_rt_fresh_asI2 [intro]:
  assumes " $\neg$  (rt2  $\sqsubseteq_{dip}$  rt1)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
proof
  assume "rt1  $\approx_{dip}$  rt2"
  hence "rt2  $\sqsubseteq_{dip}$  rt1" ..
  with  $\langle \neg$  (rt2  $\sqsubseteq_{dip}$  rt1)  $\rangle$  show False ..
qed

lemma not_single_rt_fresher [elim]:
  assumes " $\neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))"
  shows " $\neg$ (rt1  $\sqsubseteq_{ip}$  rt2)"
proof
  assume "rt1  $\sqsubseteq_{ip}$  rt2"
  hence "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)" ..
  with  $\langle \neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))  $\rangle$  show False ..
qed

lemmas not_single_rt_fresh_asI1 [intro] = not_rt_fresh_asI1 [OF not_single_rt_fresher]

```



lemmas not\_single\_rt\_fresh\_asI2 [intro] = not\_rt\_fresh\_asI2 [OF not\_single\_rt\_fresher]

lemma not\_rt\_fresher\_single [elim]:  
 assumes " $\neg(rt1 \sqsubseteq_{ip} rt2)$ "  
 shows " $\neg(\text{the } (rt1 \text{ ip}) \sqsubseteq \text{the } (rt2 \text{ ip}))$ "  
 proof  
 assume " $\text{the } (rt1 \text{ ip}) \sqsubseteq \text{the } (rt2 \text{ ip})$ "  
 hence " $rt1 \sqsubseteq_{ip} rt2$ " ..  
 with " $\neg(rt1 \sqsubseteq_{ip} rt2)$ " show False ..  
 qed

lemma rt\_fresh\_as\_nsqnr:  
 assumes " $dip \in kD(rt1)$ "  
 and " $dip \in kD(rt2)$ "  
 and " $rt1 \approx_{dip} rt2$ "  
 shows " $nsqn_r (\text{the } (rt2 \text{ dip})) = nsqn_r (\text{the } (rt1 \text{ dip}))$ "  
 using assms(3) unfolding rt\_fresh\_as\_def  
 by (auto simp: rt\_fresher\_def2 [OF  $\langle dip \in kD(rt1) \rangle \langle dip \in kD(rt2) \rangle$ ]  
 rt\_fresher\_def2 [OF  $\langle dip \in kD(rt2) \rangle \langle dip \in kD(rt1) \rangle$ ]  
 kD\_nsqn [OF  $\langle dip \in kD(rt1) \rangle$ ]  
 kD\_nsqn [OF  $\langle dip \in kD(rt2) \rangle$ ])

lemma rt\_fresher\_mapupd [intro!]:  
 assumes " $dip \in kD(rt)$ "  
 and " $\text{the } (rt \text{ dip}) \sqsubseteq r$ "  
 shows " $rt \sqsubseteq_{dip} rt(dip \mapsto r)$ "  
 using assms unfolding rt\_fresher\_def by simp

lemma rt\_fresher\_map\_update\_other [intro!]:  
 assumes " $dip \in kD(rt)$ "  
 and " $dip \neq ip$ "  
 shows " $rt \sqsubseteq_{dip} rt(ip \mapsto r)$ "  
 using assms unfolding rt\_fresher\_def by simp

lemma rt\_fresher\_update\_other [simp]:  
 assumes inkD: " $dip \in kD(rt)$ "  
 and " $dip \neq ip$ "  
 shows " $rt \sqsubseteq_{dip} \text{update } rt \text{ ip } r$ "  
 using assms unfolding update\_def  
 by (clarsimp split: option.split) (fastforce)

theorem rt\_fresher\_update [simp]:  
 assumes " $dip \in kD(rt)$ "  
 and " $\text{the } (dhops \text{ rt } dip) \geq 1$ "  
 and " $\text{update\_arg\_wf } r$ "  
 shows " $rt \sqsubseteq_{dip} \text{update } rt \text{ ip } r$ "  
 proof (cases " $dip = ip$ ")  
 assume " $dip \neq ip$ " with  $\langle dip \in kD(rt) \rangle$  show ?thesis  
 by (rule rt\_fresher\_update\_other)  
 next  
 assume " $dip = ip$ "  
 from  $\langle dip \in kD(rt) \rangle$  obtain  $dsn_n \ dsk_n \ f_n \ hops_n \ nhip_n \ pre_n$   
 where  $rtn$  [simp]: " $\text{the } (rt \text{ dip}) = (dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)$ "  
 by (metis prod\_cases6)  
 with  $\langle \text{the } (dhops \text{ rt } dip) \geq 1 \rangle$  and  $\langle dip \in kD(rt) \rangle$  have " $hops_n \geq 1$ "  
 by (metis proj5\_eq\_dhops projs(4))  
 from  $\langle dip \in kD(rt) \rangle \text{ rtn}$  have [simp]: " $sqn \text{ rt } dip = dsn_n$ "  
 and [simp]: " $\text{the } (dhops \text{ rt } dip) = hops_n$ "  
 and [simp]: " $\text{the } (flag \text{ rt } dip) = f_n$ "  
 by (simp add: sqn\_def proj5\_eq\_dhops [symmetric]  
 proj4\_eq\_flag [symmetric])  
 from  $\langle \text{update\_arg\_wf } r \rangle$  have " $(dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)$ "

```

       $\sqsubseteq$  the ((update rt dip r) dip)"
proof (rule wf_r_cases)
  fix nhip pre
  from (hopsn ≥ 1) have " $\wedge$ pre'. (dsnn, dskn, fn, hopsn, nhipn, pren)
     $\sqsubseteq$  (dsnn, unk, val, Suc 0, nhip, pre')"
  unfolding fresher_def sqn_def by (cases fn) auto
  thus "(dsnn, dskn, fn, hopsn, nhipn, pren)
     $\sqsubseteq$  the (update rt dip (0, unk, val, Suc 0, nhip, pre) dip)"
  using (dip ∈ kD(rt)) by - (rule update_cases_kD, simp_all)
next
fix dsn :: sqn and hops nhip pre
assume "0 < dsn"
show "(dsnn, dskn, fn, hopsn, nhipn, pren)
   $\sqsubseteq$  the (update rt dip (dsn, kno, val, hops, nhip, pre) dip)"
proof (rule update_cases_kD [OF _ (dip ∈ kD(rt))], simp_all add: (0 < dsn))
  assume "dsnn < dsn"
  thus "(dsnn, dskn, fn, hopsn, nhipn, pren)
     $\sqsubseteq$  (dsn, kno, val, hops, nhip, pre  $\cup$  pren)"
  unfolding fresher_def by auto
next
  assume "dsnn = dsn"
  and "hops < hopsn"
  thus "(dsn, dskn, fn, hopsn, nhipn, pren)
     $\sqsubseteq$  (dsn, kno, val, hops, nhip, pre  $\cup$  pren)"
  unfolding fresher_def nsqnr_def by simp
next
  assume "dsnn = dsn"
  with (0 < dsn)
  show "(dsn, dskn, inv, hopsn, nhipn, pren)
     $\sqsubseteq$  (dsn, kno, val, hops, nhip, pre  $\cup$  pren)"
  unfolding fresher_def by simp
qed
qed
hence "rt  $\sqsubseteq_{dip}$  update rt dip r"
  by - (rule single_rt_fresher, simp)
with (dip = ip) show ?thesis by simp
qed

theorem rt_fresher_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
  and indests: " $\forall$ rip ∈ dom(dests). rip ∈ vD(rt)  $\wedge$  sqn rt rip < the (dests rip)"
  shows "rt  $\sqsubseteq_{dip}$  invalidate rt dests"
proof (cases "dip ∈ dom(dests)")
  assume "dip ∉ dom(dests)"
  thus ?thesis using (dip ∈ kD(rt))
  by - (rule single_rt_fresher, simp)
next
  assume "dip ∈ dom(dests)"
  moreover with indests have "dip ∈ vD(rt)"
  and "sqn rt dip < the (dests dip)"
  by auto
  ultimately show ?thesis
  unfolding invalidate_def sqn_def
  by - (rule single_rt_fresher, auto simp: fresher_def)
qed

lemma nsqnr_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
  and "dip ∈ dom(dests)"
  shows "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
  using assms unfolding invalidate_def by auto

lemma rt_fresh_as_inc_invalidate [simp]:
  assumes "dip ∈ kD(rt)"

```

```

    and "∀rip∈dom(dests). rip∈vD(rt) ∧ the (dests rip) = inc (sqn rt rip)"
  shows "rt ≈dip invalidate rt dests"
proof (cases "dip∈dom(dests)")
  assume "dip∉dom(dests)"
  with ⟨dip∈kD(rt)⟩ have "dip∈kD(invalidate rt dests)"
    by simp
  with ⟨dip∈kD(rt)⟩ show ?thesis
    by rule (simp_all add: ⟨dip∉dom(dests)⟩)
next
  assume "dip∈dom(dests)"
  with assms(2) have "dip∈vD(rt)"
    and "the (dests dip) = inc (sqn rt dip)" by auto
  from ⟨dip∈vD(rt)⟩ have "dip∈kD(rt)" by simp
  moreover then have "dip∈kD(invalidate rt dests)" by simp
  ultimately show ?thesis
proof (rule nsqn_rt_fresh_asI)
  from ⟨dip∈vD(rt)⟩ have "nsqn rt dip = sqn rt dip" by simp
  also have "sqn rt dip = nsqnr (the (invalidate rt dests dip))"
  proof -
    from ⟨dip∈kD(rt)⟩ have "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
      using ⟨dip∈dom(dests)⟩ by (rule nsqnr_invalidate)
    with ⟨the (dests dip) = inc (sqn rt dip)⟩
      show "sqn rt dip = nsqnr (the (invalidate rt dests dip))" by simp
  qed
  also from ⟨dip∈kD(invalidate rt dests)⟩
    have "nsqnr (the (invalidate rt dests dip)) = nsqn (invalidate rt dests) dip"
      by (simp add: kD_nsqn)
    finally show "nsqn rt dip = nsqn (invalidate rt dests) dip" .
  qed simp
qed

```

lemmas `rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]`

```

lemma rt_fresh_as_addpreRT [simp]:
  assumes "ip∈kD(rt)"
  shows "rt ≈dip the (addpreRT rt ip npre)"
  using assms [THEN kD_Some] by (auto simp: addpreRT_def)

```

lemmas `rt_fresher_addpreRT [simp] = rt_fresh_as_addpreRT [THEN rt_fresh_asD1]`

## 0.6.4 Strictly comparing routing tables

```

definition rt_strictly_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"

```

where

```

"rt_strictly_fresher ≡ λdip rt1 rt2. (rt1 ⊆dip rt2) ∧ ¬(rt1 ≈dip rt2)"

```

abbreviation

```

rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊆  _)" [51, 999, 51] 50)

```

where

```

"rt1 ⊆i rt2 ≡ rt_strictly_fresher i rt1 rt2"

```

```

lemma rt_strictly_fresher_def'':

```

```

"rt1 ⊆i rt2 = ((rt1 ⊆i rt2) ∧ ¬(rt2 ⊆i rt1))"
  unfolding rt_strictly_fresher_def rt_fresh_as_def by auto

```

```

lemma rt_strictly_fresherI' [intro]:

```

```

  assumes "rt1 ⊆i rt2"
    and "¬(rt2 ⊆i rt1)"
  shows "rt1 ⊆i rt2"
  using assms unfolding rt_strictly_fresher_def'' by simp

```

```

lemma rt_strictly_fresherE' [elim]:

```

```

  assumes "rt1 ⊆i rt2"
    and "[| rt1 ⊆i rt2; ¬(rt2 ⊆i rt1) |] ⇒ P rt1 rt2 i"

```

```

shows "P rt1 rt2 i"
using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherI [intro]:
  assumes "rt1  $\sqsubseteq_i$  rt2"
    and " $\neg(rt1 \approx_i rt2)$ "
  shows "rt1  $\sqsubseteq_i$  rt2"
  unfolding rt_strictly_fresher_def using assms ..

lemmas rt_strictly_fresher_singleI [elim] = rt_strictly_fresherI [OF single_rt_fresher]

lemma rt_strictly_fresherE [elim]:
  assumes "rt1  $\sqsubseteq_i$  rt2"
    and "[[ rt1  $\sqsubseteq_i$  rt2;  $\neg(rt1 \approx_i rt2)$  ] ]  $\implies$  P rt1 rt2 i"
  shows "P rt1 rt2 i"
  using assms(1) unfolding rt_strictly_fresher_def
  by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':
  "rt1  $\sqsubseteq_i$  rt2 =
  (nsqnr (the (rt1 i)) < nsqnr (the (rt2 i))
   $\vee$  (nsqnr (the (rt1 i)) = nsqnr (the (rt2 i))  $\wedge$   $\pi_5$ (the (rt1 i)) >  $\pi_5$ (the (rt2 i))))"
  unfolding rt_strictly_fresher_def'' rt_fresher_def fresher_def by auto

lemma rt_strictly_fresher_fresherD [dest]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  shows "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"
  using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  shows " $\neg rt1 \approx_{dip} rt2$ "
  using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
  using assms proof -
    from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  obtain "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)" by auto
    also from  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  obtain "the (rt2 dip)  $\sqsubseteq$  the (rt3 dip)" by auto
    finally have "the (rt1 dip)  $\sqsubseteq$  the (rt3 dip)" .

    moreover have " $\neg (rt1 \approx_{dip} rt3)$ "
  proof -
    from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  obtain " $\neg$ (the (rt2 dip)  $\sqsubseteq$  the (rt1 dip))" by auto
    also from  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  obtain " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt2 dip))" by auto
    finally have " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt1 dip))" .
    thus ?thesis ..
  qed
  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: " $\neg (rt \sqsubseteq_{dip} rt)$ "
  using assms unfolding rt_strictly_fresher_def
  by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
  proof -
    from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  have "rt1  $\sqsubseteq_{dip}$  rt2"
      and " $\neg$ (rt2  $\sqsubseteq_{dip}$  rt1)"

```

```

    unfolding rt_strictly_fresher_def'' by auto
  from this(1) and ⟨rt2  $\sqsubseteq_{dip}$  rt3⟩ have "rt1  $\sqsubseteq_{dip}$  rt3" ..

  moreover from ⟨¬(rt2  $\sqsubseteq_{dip}$  rt1)⟩ have "¬(rt3  $\sqsubseteq_{dip}$  rt1)"
  proof (rule contrapos_nn)
    assume "rt3  $\sqsubseteq_{dip}$  rt1"
    with ⟨rt2  $\sqsubseteq_{dip}$  rt3⟩ show "rt2  $\sqsubseteq_{dip}$  rt1" ..
  qed

  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3"
  unfolding rt_strictly_fresher_def'' by auto
qed

lemma rt_fresher_trans_rt_strictly_fresher' [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
proof -
  from ⟨rt2  $\sqsubseteq_{dip}$  rt3⟩ have "rt2  $\sqsubseteq_{dip}$  rt3"
    and "¬(rt3  $\sqsubseteq_{dip}$  rt2)"
  unfolding rt_strictly_fresher_def'' by auto
  from ⟨rt1  $\sqsubseteq_{dip}$  rt2⟩ and this(1) have "rt1  $\sqsubseteq_{dip}$  rt3" ..

  moreover from ⟨¬(rt3  $\sqsubseteq_{dip}$  rt2)⟩ have "¬(rt3  $\sqsubseteq_{dip}$  rt1)"
  proof (rule contrapos_nn)
    assume "rt3  $\sqsubseteq_{dip}$  rt1"
    thus "rt3  $\sqsubseteq_{dip}$  rt2" using ⟨rt1  $\sqsubseteq_{dip}$  rt2⟩ ..
  qed

  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3"
  unfolding rt_strictly_fresher_def'' by auto
qed

lemma rt_fresher_imp_nsqn_le:
  assumes "rt1  $\sqsubseteq_{ip}$  rt2"
    and "ip  $\in$  kD rt1"
    and "ip  $\in$  kD rt2"
  shows "nsqn rt1 ip  $\leq$  nsqn rt2 ip"
using assms(1)
by (auto simp add: rt_fresher_def2 [OF assms(2-3)])

lemma rt_strictly_fresher_ltI [intro]:
  assumes "dip  $\in$  kD(rt1)"
    and "dip  $\in$  kD(rt2)"
    and "nsqn rt1 dip < nsqn rt2 dip"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
proof
  from assms show "rt1  $\sqsubseteq_{dip}$  rt2" ..
next
  show "¬(rt1  $\approx_{dip}$  rt2)"
  proof
    assume "rt1  $\approx_{dip}$  rt2"
    hence "rt2  $\sqsubseteq_{dip}$  rt1" ..
    hence "nsqn rt2 dip  $\leq$  nsqn rt1 dip"
      using ⟨dip  $\in$  kD(rt2)⟩ ⟨dip  $\in$  kD(rt1)⟩
      by (rule rt_fresher_imp_nsqn_le)
    with ⟨nsqn rt1 dip < nsqn rt2 dip⟩ show "False"
      by simp
  qed
qed
qed

lemma rt_strictly_fresher_eqI [intro]:
  assumes "i  $\in$  kD(rt1)"
    and "i  $\in$  kD(rt2)"

```

```

    and "nsqn rt1 i = nsqn rt2 i"
    and " $\pi_5(\text{the } (rt2 \ i)) < \pi_5(\text{the } (rt1 \ i))$ "
    shows "rt1  $\sqsubset_i$  rt2"
using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

```

```

lemma invalidate_rtsf_left [simp]:
  " $\bigwedge \text{dests dip rt rt}'. \text{dests dip} = \text{None} \implies (\text{invalidate rt dests } \sqsubset_{\text{dip}} \text{rt}') = (\text{rt } \sqsubset_{\text{dip}} \text{rt}')$ "
  unfolding invalidate_def rt_strictly_fresher_def'
  by (rule iffI) (auto split: option.split_asm)

```

```

lemma vD_invalidate_rt_strictly_fresher [simp]:
  assumes "dip  $\in$  vD(invalidate rt1 dests)"
  shows "(invalidate rt1 dests  $\sqsubset_{\text{dip}}$  rt2) = (rt1  $\sqsubset_{\text{dip}}$  rt2)"
  proof (cases "dip  $\in$  dom(dests)")
    assume "dip  $\in$  dom(dests)"
    hence "dip  $\notin$  vD(invalidate rt1 dests)"
      unfolding invalidate_def vD_def
      by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
    with  $\langle \text{dip} \in \text{vD}(\text{invalidate rt1 dests}) \rangle$  show ?thesis by simp
  next
    assume "dip  $\notin$  dom(dests)"
    hence "dests dip = None" by auto
    moreover with  $\langle \text{dip} \in \text{vD}(\text{invalidate rt1 dests}) \rangle$  have "dip  $\in$  vD(rt1)"
      unfolding invalidate_def vD_def
      by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
    ultimately show ?thesis
      unfolding invalidate_def rt_strictly_fresher_def' by auto
  qed

```

```

lemma rt_strictly_fresher_update_other [elim!]:
  " $\bigwedge \text{dip ip rt r rt}'. [\text{dip} \neq \text{ip}; \text{rt } \sqsubset_{\text{dip}} \text{rt}'] \implies \text{update rt ip r } \sqsubset_{\text{dip}} \text{rt}'$ "
  unfolding rt_strictly_fresher_def' by clarsimp

```

```

lemma addpreRT_strictly_fresher [simp]:
  assumes "dip  $\in$  kD(rt)"
  shows "(the (addpreRT rt dip npre)  $\sqsubset_{\text{ip}}$  rt2) = (rt  $\sqsubset_{\text{ip}}$  rt2)"
  using assms unfolding rt_strictly_fresher_def' by clarsimp

```

```

lemma lt_sqn_imp_update_strictly_fresher:
  assumes "dip  $\in$  vD (rt2 nhip)"
    and *: "osn < sqn (rt2 nhip) dip"
    and **: "rt  $\neq$  update rt dip (osn, kno, val, hops, nhip, {})"
  shows "update rt dip (osn, kno, val, hops, nhip, {})  $\sqsubset_{\text{dip}}$  rt2 nhip"
  unfolding rt_strictly_fresher_def'
  proof (rule disjI1)
    from ** have "nsqn (update rt dip (osn, kno, val, hops, nhip, {})) dip = osn"
      by (rule nsqn_update_changed_kno_val)
    with  $\langle \text{dip} \in \text{vD}(\text{rt2 nhip}) \rangle$ 
      have "nsqnr (the (update rt dip (osn, kno, val, hops, nhip, {}) dip)) = osn"
        by (simp add: kD_nsqn)
    also have "osn < sqn (rt2 nhip) dip" by (rule *)
    also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip dip))"
      unfolding nsqnr_def using  $\langle \text{dip} \in \text{vD} (\text{rt2 nhip}) \rangle$ 
      by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
    finally show "nsqnr (the (update rt dip (osn, kno, val, hops, nhip, {}) dip))
      < nsqnr (the (rt2 nhip dip))" .
  qed

```

```

lemma dhops_le_hops_imp_update_strictly_fresher:
  assumes "dip  $\in$  vD(rt2 nhip)"
    and sqn: "sqn (rt2 nhip) dip = osn"
    and hop: "the (dhops (rt2 nhip) dip)  $\leq$  hops"
    and **: "rt  $\neq$  update rt dip (osn, kno, val, Suc hops, nhip, {})"
  shows "update rt dip (osn, kno, val, Suc hops, nhip, {})  $\sqsubset_{\text{dip}}$  rt2 nhip"

```

```

unfolding rt_strictly_fresher_def'
proof (rule disjI2, rule conjI)
  from ** have "nsqn (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip = osn"
    by (rule nsqn_update_changed_kno_val)
  with ⟨dip ∈ vD(rt2 nhip)⟩
    have "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip)) = osn"
      by (simp add: kD_nsqn)
  also have "osn = sqn (rt2 nhip) dip" by (rule sqn [symmetric])
  also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip) dip)"
    unfolding nsqnr_def using ⟨dip ∈ vD(rt2 nhip)⟩
    by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip))
    = nsqnr (the (rt2 nhip) dip))" .

next
  have "the (dhops (rt2 nhip) dip) ≤ hops" by (rule hop)
  also have "hops < hops + 1" by simp
  also have "hops + 1 = the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip))"
    using ** by simp
  finally have "the (dhops (rt2 nhip) dip)
    < the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip))" .
  thus "π5 (the (rt2 nhip) dip) < π5 (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip))"
    using ⟨dip ∈ vD(rt2 nhip)⟩ by (simp add: proj5_eq_dhops)
qed

```

```

lemma nsqn_invalidate:
  assumes "dip ∈ kD(rt)"
    and "∀ip ∈ dom(dests). ip ∈ vD(rt) ∧ the (dests ip) = inc (sqn rt ip)"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
proof -
  from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp

  from assms have "rt ≈dip invalidate rt dests"
    by (rule rt_fresh_as_inc_invalidate)
  with ⟨dip ∈ kD(rt)⟩ ⟨dip ∈ kD(invalidate rt dests)⟩ show ?thesis
    by (simp add: kD_nsqn del: invalidate_kD_inv)
      (erule(2) rt_fresh_as_nsqnr)
qed

```

end

## 0.7 Invariant proofs on individual processes

```

theory Seq_Invariants
imports "../AWN/Invariants" Aadv Aadv_Data Aadv_Predicates Fresher

```

begin

The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

```

lemma sequence_number_increases:
  "paadv i ⊨A onll ΓAODV (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
  by inv_cterms

```

```

lemma sequence_number_one_or_bigger:
  "paadv i ⊨ onl ΓAODV (λ(ξ, _). 1 ≤ sn ξ)"
  by (rule onll_step_to_invariantI [OF sequence_number_increases])
    (auto simp: σAODV_def)

```

We can get rid of the onl/onll if desired...

```

lemma sequence_number_increases':
  "paadv i ⊨A (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
  by (rule step_invariant_weakenE [OF sequence_number_increases]) (auto dest!: onllD)

```

lemma sequence\_number\_one\_or\_bigger':

```
"paadv i ⊨ (λ(ξ, _). 1 ≤ sn ξ)"
by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto
```

lemma sip\_in\_kD:

```
"paadv i ⊨ onl ΓAODV (λ(ξ, l). l ∈ ({PAadv-:7} ∪ {PAadv-:5} ∪ {PRrep-:0..PRrep-:1}
    ∪ {PRreq-:0..PRreq-:3}) → sip ξ ∈ kD (rt ξ))"
by inv_cterms
```

lemma rrep\_1\_update\_changes:

```
"paadv i ⊨ onl ΓAODV (λ(ξ, l). (l = PRrep-:1 →
    rt ξ ≠ update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, { })))"
by inv_cterms
```

lemma addpreRT\_partly\_welldefined:

```
"paadv i ⊨
    onl ΓAODV (λ(ξ, l). (l ∈ {PRreq-:16..PRreq-:18} ∪ {PRrep-:2..PRrep-:6} → dip ξ ∈ kD (rt ξ))
    ∧ (l ∈ {PRreq-:3..PRreq-:17} → oip ξ ∈ kD (rt ξ)))"
by inv_cterms
```

Proposition 7.38

lemma includes\_nhip:

```
"paadv i ⊨ onl ΓAODV (λ(ξ, l). ∀dip∈kD(rt ξ). the (nhop (rt ξ) dip)∈kD(rt ξ))"
```

proof -

```
{ fix ip and ξ ξ' :: state
  assume "∀dip∈kD (rt ξ). the (nhop (rt ξ) dip) ∈ kD (rt ξ)"
  and "ξ' = ξ(|rt := update (rt ξ) ip (0, unk, val, Suc 0, ip, { })|)"
  hence "∀dip∈kD (rt ξ).
    the (nhop (update (rt ξ) ip (0, unk, val, Suc 0, ip, { })) dip) = ip
    ∨ the (nhop (update (rt ξ) ip (0, unk, val, Suc 0, ip, { })) dip) ∈ kD (rt ξ)"
  by clarsimp (metis nhop_update_unk_val update_another)
} note one_hop = this
{ fix ip sip sn hops and ξ ξ' :: state
  assume "∀dip∈kD (rt ξ). the (nhop (rt ξ) dip) ∈ kD (rt ξ)"
  and "ξ' = ξ(|rt := update (rt ξ) ip (sn, kno, val, Suc hops, sip, { })|)"
  and "sip ∈ kD (rt ξ)"
  hence "(the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, { })) ip) = ip
    ∨ the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, { })) ip) ∈ kD (rt ξ))
    ∧ (∀dip∈kD (rt ξ).
      the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, { })) dip) = ip
      ∨ the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, { })) dip) ∈ kD (rt ξ))"
  by (metis kD_update_unchanged nhop_update_changed update_another)
} note nhip_is_sip = this
show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf sip_in_kD]
    onl_invariant_sterms [OF aadv_wf addpreRT_partly_welldefined]
    solve: one_hop nhip_is_sip)
```

qed

Proposition 7.22: needed in Proposition 7.4

lemma addpreRT\_welldefined:

```
"paadv i ⊨ onl ΓAODV (λ(ξ, l). (l ∈ {PRreq-:16..PRreq-:18} → dip ξ ∈ kD (rt ξ)) ∧
    (l = PRreq-:17 → oip ξ ∈ kD (rt ξ)) ∧
    (l = PRrep-:5 → dip ξ ∈ kD (rt ξ)) ∧
    (l = PRrep-:6 → (the (nhop (rt ξ) (dip ξ))) ∈ kD (rt ξ)))"
```

(is "\_ ⊨ onl Γ<sub>AODV</sub> ?P")

unfolding invariant\_def

proof

```
fix s
assume "s ∈ reachable (paadv i) TT"
then obtain ξ p where "s = (ξ, p)"
and "(ξ, p) ∈ reachable (paadv i) TT"
by (metis PairE)
```



```

have "onl  $\Gamma_{AODV}$  ?P ( $\xi$ , p)"
proof (rule onlI)
  fix l
  assume "l  $\in$  labels  $\Gamma_{AODV}$  p"
  with  $\langle (\xi, p) \in \text{reachable (paodv i) TT} \rangle$ 
    have I1: "l  $\in$  {PRreq-:16..PRreq-:18}  $\longrightarrow$  dip  $\xi \in \text{kD}(\text{rt } \xi)$ "
    and I2: "l = PRreq-:17  $\longrightarrow$  oip  $\xi \in \text{kD}(\text{rt } \xi)$ "
    and I3: "l  $\in$  {PRrep-:2..PRrep-:6}  $\longrightarrow$  dip  $\xi \in \text{kD}(\text{rt } \xi)$ "
    by (auto dest!: invariantD [OF addpreRT_partly_welldefined])
  moreover from  $\langle (\xi, p) \in \text{reachable (paodv i) TT} \rangle$   $\langle l \in \text{labels } \Gamma_{AODV} p \rangle$  and I3
    have "l = PRrep-:6  $\longrightarrow$  (the (nhop (rt  $\xi$ ) (dip  $\xi$ )))  $\in \text{kD}(\text{rt } \xi)$ "
    by (auto dest!: invariantD [OF includes_nhip])
  ultimately show "?P ( $\xi$ , l)"
    by simp
qed
with  $\langle s = (\xi, p) \rangle$  show "onl  $\Gamma_{AODV}$  ?P s"
  by simp
qed

```

Proposition 7.4

```

lemma known_destinations_increase:
  "paodv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_)). \text{kD}(\text{rt } \xi) \subseteq \text{kD}(\text{rt } \xi')$ )"
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]
      simp add: subset_insertI)

```

Proposition 7.5

```

lemma rreqs_increase:
  "paodv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_)). \text{rreqs } \xi \subseteq \text{rreqs } \xi')$ "
  by (inv_cterms simp add: subset_insertI)

```

lemma dests\_bigger\_than\_sqn:

```

"paodv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l). l \in \{\text{PAodv-:15..PAodv-:19}\} \cup \{\text{PPkt-:7..PPkt-:11}\} \cup \{\text{PRreq-:9..PRreq-:13}\} \cup \{\text{PRreq-:21..PRreq-:25}\} \cup \{\text{PRrep-:10..PRrep-:14}\} \cup \{\text{PRerr-:1..PRerr-:5}\} \longrightarrow (\forall ip \in \text{dom}(\text{dests } \xi). ip \in \text{kD}(\text{rt } \xi) \wedge \text{sqn}(\text{rt } \xi) ip \leq \text{the}(\text{dests } \xi ip))$ )"

```

proof -

```

have sqninv:
  " $\bigwedge \text{dests rt rsn ip. } [\forall ip \in \text{dom}(\text{dests}). ip \in \text{kD}(\text{rt}) \wedge \text{sqn rt ip} \leq \text{the}(\text{dests ip}); \text{dests ip} = \text{Some rsn}] \implies \text{sqn}(\text{invalidate rt dests}) ip \leq \text{rsn}$ "
  by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto
have indests:
  " $\bigwedge \text{dests rt rsn ip. } [\forall ip \in \text{dom}(\text{dests}). ip \in \text{kD}(\text{rt}) \wedge \text{sqn rt ip} \leq \text{the}(\text{dests ip}); \text{dests ip} = \text{Some rsn}] \implies ip \in \text{kD}(\text{rt}) \wedge \text{sqn rt ip} \leq \text{rsn}$ "
  by (metis domI option.sel)
show ?thesis
  by inv_cterms
  (clarsimp split: split_if_asm option.split_asm
   elim!: sqninv indests)+

```

qed

Proposition 7.6

```

lemma sqns_increase:
  "paodv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_)). \forall ip. \text{sqn}(\text{rt } \xi) ip \leq \text{sqn}(\text{rt } \xi') ip$ )"
proof -
  { fix  $\xi :: \text{state}$ 
    assume *: " $\forall ip \in \text{dom}(\text{dests } \xi). ip \in \text{kD}(\text{rt } \xi) \wedge \text{sqn}(\text{rt } \xi) ip \leq \text{the}(\text{dests } \xi ip)$ "
    have " $\forall ip. \text{sqn}(\text{rt } \xi) ip \leq \text{sqn}(\text{invalidate}(\text{rt } \xi)(\text{dests } \xi)) ip$ "
    proof
      fix ip

```

```

from * have "ip $\notin$ dom(dests  $\xi$ )  $\vee$  sqn (rt  $\xi$ ) ip  $\leq$  the (dests  $\xi$  ip)" by simp
thus "sqn (rt  $\xi$ ) ip  $\leq$  sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) ip"
  by (metis domI invalidate_sqn option.sel)

```

qed

```

} note solve_invalidate = this

```

```

show ?thesis

```

```

by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf addpreRT_welldefined]
    onl_invariant_sterms [OF aadv_wf dests_bigger_than_sqn]
    simp add: solve_invalidate)

```

qed

Proposition 7.7

lemma ip\_constant:

```

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_).$  ip  $\xi$  = i)"
by (inv_cterms simp add:  $\sigma_{AODV\_def}$ )

```

Proposition 7.8

lemma sender\_ip\_valid':

```

"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_).$  anycast ( $\lambda m.$  not_Pkt m  $\longrightarrow$  msg_sender m = ip  $\xi$ ) a)"
by inv_cterms

```

lemma sender\_ip\_valid:

```

"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_).$  anycast ( $\lambda m.$  not_Pkt m  $\longrightarrow$  msg_sender m = i) a)"
by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid'])
  (auto dest!: onlD onllD)

```

lemma received\_msg\_inv:

```

"paadv i  $\models$  (recvmg P  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l).$  l  $\in$  {PAadv-:1}  $\longrightarrow$  P (msg  $\xi$ ))"
by inv_cterms

```

Proposition 7.9

lemma sip\_not\_ip':

```

"paadv i  $\models$  (recvmg ( $\lambda m.$  not_Pkt m  $\longrightarrow$  msg_sender m  $\neq$  i)  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_).$  sip  $\xi$   $\neq$  ip  $\xi$ )"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]
    onl_invariant_sterms [OF aadv_wf ip_constant [THEN invariant_restrict_inD]]
    simp add: clear_locals_sip_not_ip') clarsimp+

```

lemma sip\_not\_ip:

```

"paadv i  $\models$  (recvmg ( $\lambda m.$  not_Pkt m  $\longrightarrow$  msg_sender m  $\neq$  i)  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_).$  sip  $\xi$   $\neq$  i)"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]
    onl_invariant_sterms [OF aadv_wf ip_constant [THEN invariant_restrict_inD]]
    simp add: clear_locals_sip_not_ip') clarsimp+

```

Neither sip\_not\_ip' nor sip\_not\_ip is needed to show loop freedom.

Proposition 7.10

lemma hop\_count\_positive:

```

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_).$   $\forall ip \in kD$  (rt  $\xi$ ). the (dhops (rt  $\xi$ ) ip)  $\geq$  1)"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf addpreRT_welldefined]) auto

```

lemma rreq\_dip\_in\_vD\_dip\_eq\_ip:

```

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l).$  (l  $\in$  {PRreq-:16..PRreq-:18}  $\longrightarrow$  dip  $\xi$   $\in$  vD(rt  $\xi$ ))
   $\wedge$  (l  $\in$  {PRreq-:5, PRreq-:6}  $\longrightarrow$  dip  $\xi$  = ip  $\xi$ )
   $\wedge$  (l  $\in$  {PRreq-:15..PRreq-:18}  $\longrightarrow$  dip  $\xi$   $\neq$  ip  $\xi$ ))"

```

proof (inv\_cterms, elim conjE)

```

fix l  $\xi$  pp p'

```

```

assume "( $\xi, pp$ )  $\in$  reachable (paadv i) TT"

```

```

and "{PRreq-:17}  $\llbracket \lambda \xi. \xi \llbracket rt :=$  the (addpreRT (rt  $\xi$ ) (oip  $\xi$ ) {the (nhop (rt  $\xi$ ) (dip  $\xi$ ))}  $\rrbracket \rrbracket p'$ 
   $\in$  sterms  $\Gamma_{AODV}$  pp"

```

```

and "l = PRreq-:17"

```

```

and "dip  $\xi$   $\in$  vD (rt  $\xi$ )"

```

```

from this(1-3) have "oip  $\xi$   $\in$  kD (rt  $\xi$ )"

```

```

by (auto dest: onl_invariant_sterms [OF aadv_wf addpreRT_welldefined, where l="PRreq-:17"])

```

```

with ⟨dip ξ ∈ vD (rt ξ)⟩
  show "dip ξ ∈ vD (the (addpreRT (rt ξ) (oip ξ) {the (nhop (rt ξ) (dip ξ))}))" by simp
qed

```

Proposition 7.11

lemma anycast\_msg\_zhops:

```

"∧rreqid dip dsn dsk oip osn sip.
  paadv i ⊨A onll ΓAODV (λ(⟦, a, ⟦). anycast msg_zhops a)"
proof (inv_cterms inv add:
  onl_invariant_sterms [OF aadv_wf rreq_dip_in_vD_dip_eq_ip [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aadv_wf hop_count_positive [THEN invariant_restrict_inD]],
  elim conjE)
fix l ξ a pp p' pp'
assume "(ξ, pp) ∈ reachable (paadv i) TT"
  and "{PRreq-:18}unicast(λξ. the (nhop (rt ξ) (oip ξ)),
    λξ. Rrep (the (dhops (rt ξ) (dip ξ))) (dip ξ) (sqn (rt ξ) (dip ξ)) (oip ξ) (ip ξ)).
    p' ▷ pp' ∈ sterms ΓAODV pp)"
  and "l = PRreq-:18"
  and "a = unicast (the (nhop (rt ξ) (oip ξ)))
    (Rrep (the (dhops (rt ξ) (dip ξ))) (dip ξ) (sqn (rt ξ) (dip ξ)) (oip ξ) (ip ξ))"
  and *: "∀ip∈kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) ip)"
  and "dip ξ ∈ vD (rt ξ)"
from ⟨dip ξ ∈ vD (rt ξ)⟩ have "dip ξ ∈ kD (rt ξ)"
  by (rule vD_iD_gives_kD(1))
with * have "Suc 0 ≤ the (dhops (rt ξ) (dip ξ))" ..
thus "0 < the (dhops (rt ξ) (dip ξ))" by simp
qed

```

lemma hop\_count\_zero\_oip\_dip\_sip:

```

"paadv i ⊨ (recvmmsg msg_zhops →) onl ΓAODV (λ(ξ, l).
  (l ∈ {PAadv-:4..PAadv-:5} ∪ {PRreq-:n/n. True} →
    (hops ξ = 0 → oip ξ = sip ξ))
  ∧
  ((l ∈ {PAadv-:6..PAadv-:7} ∪ {PRrep-:n/n. True} →
    (hops ξ = 0 → dip ξ = sip ξ))))"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]) auto

```

lemma osn\_rreq:

```

"paadv i ⊨ (recvmmsg rreq_rrep_sn →) onl ΓAODV (λ(ξ, l).
  l ∈ {PAadv-:4, PAadv-:5} ∪ {PRreq-:n/n. True} → 1 ≤ osn ξ)"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]) clarsimp

```

lemma osn\_rreq':

```

"paadv i ⊨ (recvmmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl ΓAODV (λ(ξ, l).
  l ∈ {PAadv-:4, PAadv-:5} ∪ {PRreq-:n/n. True} → 1 ≤ osn ξ)"
proof (rule invariant_weakenE [OF osn_rreq])
  fix a
  assume "recvmmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a"
  thus "recvmmsg rreq_rrep_sn a"
    by (cases a) simp_all
qed

```

lemma dsn\_rrep:

```

"paadv i ⊨ (recvmmsg rreq_rrep_sn →) onl ΓAODV (λ(ξ, l).
  l ∈ {PAadv-:6, PAadv-:7} ∪ {PRrep-:n/n. True} → 1 ≤ dsn ξ)"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]) clarsimp

```

lemma dsn\_rrep':

```

"paadv i ⊨ (recvmmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl ΓAODV (λ(ξ, l).
  l ∈ {PAadv-:6, PAadv-:7} ∪ {PRrep-:n/n. True} → 1 ≤ dsn ξ)"
proof (rule invariant_weakenE [OF dsn_rrep])
  fix a
  assume "recvmmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a"
  thus "recvmmsg rreq_rrep_sn a"

```

by (cases a) simp\_all  
qed

lemma hop\_count\_zero\_oip\_dip\_sip':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq_rrep\_sn\ m \wedge msg\_zhops\ m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ ).  
 $(1 \in \{PAadv-:4..PAadv-:5\} \cup \{PRreq-:n|n.\ True\} \rightarrow$   
 $(hops\ \xi = 0 \rightarrow oip\ \xi = sip\ \xi))$   
 $\wedge$   
 $((1 \in \{PAadv-:6..PAadv-:7\} \cup \{PRrep-:n|n.\ True\} \rightarrow$   
 $(hops\ \xi = 0 \rightarrow dip\ \xi = sip\ \xi))))"$

proof (rule invariant\_weakenE [OF hop\_count\_zero\_oip\_dip\_sip])

fix a  
assume "recvmsg ( $\lambda m. rreq_rrep\_sn\ m \wedge msg\_zhops\ m$ ) a"  
thus "recvmsg msg\_zhops a"  
by (cases a) simp\_all  
qed

Proposition 7.12

lemma zero\_seq\_unk\_hops\_one':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq_rrep\_sn\ m \wedge msg\_zhops\ m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_)$ .  
 $\forall dip \in kD(rt\ \xi). (sqn\ (rt\ \xi)\ dip = 0 \rightarrow sqnf\ (rt\ \xi)\ dip = unk)$   
 $\wedge (sqnf\ (rt\ \xi)\ dip = unk \rightarrow the\ (dhops\ (rt\ \xi)\ dip) = 1)$   
 $\wedge (the\ (dhops\ (rt\ \xi)\ dip) = 1 \rightarrow the\ (nhop\ (rt\ \xi)\ dip) = dip))"$

proof -

{ fix dip and  $\xi :: state$  and P  
assume "sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip = 0"  
and all: " $\forall ip. sqn\ (rt\ \xi)\ ip \leq sqn\ (invalidate\ (rt\ \xi)\ (dests\ \xi))\ ip$ "  
and \*: "sqn (rt  $\xi$ ) dip = 0  $\implies$  P  $\xi$  dip"  
have "P  $\xi$  dip"  
proof -  
from all have "sqn (rt  $\xi$ ) dip  $\leq$  sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip" ..  
with (sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip = 0) have "sqn (rt  $\xi$ ) dip = 0" by simp  
thus "P  $\xi$  dip" by (rule \*)  
qed  
} note sqn\_invalidate\_zero [elim!] = this

{ fix dsn hops :: nat and sip oip rt and ip dip :: ip  
assume " $\forall dip \in kD(rt)$ .  
 $(sqn\ rt\ dip = 0 \rightarrow \pi_3(the\ (rt\ dip)) = unk) \wedge$   
 $(\pi_3(the\ (rt\ dip)) = unk \rightarrow the\ (dhops\ rt\ dip) = Suc\ 0) \wedge$   
 $(the\ (dhops\ rt\ dip) = Suc\ 0 \rightarrow the\ (nhop\ rt\ dip) = dip)"$   
and "hops = 0  $\rightarrow$  sip = dip"  
and "Suc 0  $\leq$  dsn"  
and "ip  $\neq$  dip  $\rightarrow ip \in kD(rt)"$   
hence "the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = Suc 0  $\rightarrow$   
the (nhop (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = ip"  
by - (rule update\_cases, auto simp add: sqn\_def dest!: bspec)  
} note prreq\_ok1 [simp] = this

{ fix ip dsn hops sip oip rt dip  
assume " $\forall dip \in kD(rt)$ .  
 $(sqn\ rt\ dip = 0 \rightarrow \pi_3(the\ (rt\ dip)) = unk) \wedge$   
 $(\pi_3(the\ (rt\ dip)) = unk \rightarrow the\ (dhops\ rt\ dip) = Suc\ 0) \wedge$   
 $(the\ (dhops\ rt\ dip) = Suc\ 0 \rightarrow the\ (nhop\ rt\ dip) = dip)"$   
and "Suc 0  $\leq$  dsn"  
and "ip  $\neq$  dip  $\rightarrow ip \in kD(rt)"$   
hence " $\pi_3(the\ (update\ rt\ dip\ (dsn,\ kno,\ val,\ Suc\ hops,\ sip,\ \{\})\ ip)) = unk \rightarrow$   
the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = Suc 0"  
by - (rule update\_cases, auto simp add: sqn\_def sqnf\_def dest!: bspec)  
} note prreq\_ok2 [simp] = this

{ fix ip dsn hops sip oip rt dip  
assume " $\forall dip \in kD(rt)$ .  
 $(sqn\ rt\ dip = 0 \rightarrow \pi_3(the\ (rt\ dip)) = unk) \wedge$

```

      ( $\pi_3(\text{the } (rt \text{ dip})) = \text{unk} \longrightarrow \text{the } (dhops \text{ rt dip}) = \text{Suc } 0) \wedge$ 
      ( $\text{the } (dhops \text{ rt dip}) = \text{Suc } 0 \longrightarrow \text{the } (nhop \text{ rt dip}) = \text{dip}$ )"
    and "Suc 0  $\leq$  dsn"
    and "ip  $\neq$  dip  $\longrightarrow$  ip  $\in$  kD(rt)"
  hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip, { })) ip = 0  $\longrightarrow$ 
         $\pi_3(\text{the } (\text{update rt dip } (dsn, kno, val, Suc hops, sip, { }) \text{ ip})) = \text{unk}$ "
    by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
} note prreq_ok3 [simp] = this

```

```

{ fix rt sip
  assume " $\forall$  dip  $\in$  kD rt.
    (sqn rt dip = 0  $\longrightarrow$   $\pi_3(\text{the } (rt \text{ dip})) = \text{unk}$ )  $\wedge$ 
    ( $\pi_3(\text{the } (rt \text{ dip})) = \text{unk} \longrightarrow \text{the } (dhops \text{ rt dip}) = \text{Suc } 0$ )  $\wedge$ 
    ( $\text{the } (dhops \text{ rt dip}) = \text{Suc } 0 \longrightarrow \text{the } (nhop \text{ rt dip}) = \text{dip}$ )"
  hence " $\forall$  dip  $\in$  kD rt.
    (sqn (update rt sip (0, unk, val, Suc 0, sip, { })) dip = 0  $\longrightarrow$ 
       $\pi_3(\text{the } (\text{update rt sip } (0, unk, val, Suc 0, sip, { }) \text{ dip})) = \text{unk}$ )
     $\wedge$  ( $\pi_3(\text{the } (\text{update rt sip } (0, unk, val, Suc 0, sip, { }) \text{ dip})) = \text{unk} \longrightarrow$ 
       $\text{the } (dhops (\text{update rt sip } (0, unk, val, Suc 0, sip, { }) \text{ dip}) = \text{Suc } 0)$ )
     $\wedge$  ( $\text{the } (dhops (\text{update rt sip } (0, unk, val, Suc 0, sip, { }) \text{ dip}) = \text{Suc } 0 \longrightarrow$ 
       $\text{the } (nhop (\text{update rt sip } (0, unk, val, Suc 0, sip, { }) \text{ dip}) = \text{dip})$ )"
    by - (rule update_cases, simp_all add: sqnf_def sqn_def)
} note prreq_ok4 [simp] = this

```

```

have prreq_ok5 [simp]: " $\wedge$  sip rt.
   $\pi_3(\text{the } (\text{update rt sip } (0, unk, val, Suc 0, sip, { }) \text{ sip})) = \text{unk} \longrightarrow$ 
   $\text{the } (dhops (\text{update rt sip } (0, unk, val, Suc 0, sip, { }) \text{ sip}) = \text{Suc } 0$ "
  by (rule update_cases) simp_all

```

```

have prreq_ok6 [simp]: " $\wedge$  sip rt.
  sqn (update rt sip (0, unk, val, Suc 0, sip, { })) sip = 0  $\longrightarrow$ 
   $\pi_3(\text{the } (\text{update rt sip } (0, unk, val, Suc 0, sip, { }) \text{ sip})) = \text{unk}$ "
  by (rule update_cases) simp_all

```

```

show ?thesis
  by (inv_terms inv add: onl_invariant_sterms_TT [OF aadv_wf addpreRT_welldefined]
      onl_invariant_sterms [OF aadv_wf hop_count_zero_oip_dip_sip']
      seq_step_invariant_sterms_TT [OF sqns_increase aadv_wf aadv_trans]
      onl_invariant_sterms [OF aadv_wf osn_rreq']
      onl_invariant_sterms [OF aadv_wf dsn_rrep']) clarsimp+

```

qed

lemma zero\_seq\_unk\_hops\_one:

```

"paadv i  $\models$  (recvmmsg ( $\lambda m. \text{rreq\_rrep\_sn } m \wedge \text{msg\_zhops } m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV} (\lambda(\xi, \_).$ 
   $\forall$  dip  $\in$  kD(rt  $\xi$ ). (sqn (rt  $\xi$ ) dip = 0  $\longrightarrow$  (sqnf (rt  $\xi$ ) dip = unk
     $\wedge$   $\text{the } (dhops \text{ (rt } \xi) \text{ dip}) = 1$ 
     $\wedge$   $\text{the } (nhop \text{ (rt } \xi) \text{ dip}) = \text{dip}))$ )"
  by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

```

lemma kD\_unk\_or\_atleast\_one:

```

"paadv i  $\models$  (recvmmsg rreq_rrep_sn  $\rightarrow$ ) onl  $\Gamma_{AODV} (\lambda(\xi, 1).$ 
   $\forall$  dip  $\in$  kD(rt  $\xi$ ).  $\pi_3(\text{the } (rt \text{ } \xi \text{ dip})) = \text{unk} \vee 1 \leq \pi_2(\text{the } (rt \text{ } \xi \text{ dip}))$ )"

```

proof -

```

{ fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
  assume "dsk1 = unk  $\vee$  Suc 0  $\leq$  dsn2"
  hence " $\pi_3(\text{the } (\text{update rt sip } (dsn1, dsk1, flag1, hops1, nhip1, pre1) \text{ sip})) = \text{unk}$ 
     $\vee$  Suc 0  $\leq$  sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2)) sip"
    unfolding update_def by (cases "dsk1 =unk") (clarsimp split: option.split)+
} note fromsip [simp] = this

```

```

{ fix dip sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
  assume allkd: " $\forall$  dip  $\in$  kD(rt).  $\pi_3(\text{the } (rt \text{ dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn rt dip}$ "
  and **: "dsk1 = unk  $\vee$  Suc 0  $\leq$  dsn2"
  have " $\forall$  dip  $\in$  kD(rt).  $\pi_3(\text{the } (\text{update rt sip } (dsn1, dsk1, flag1, hops1, nhip1, pre1) \text{ dip})) = \text{unk}$ "

```

```

    ∨ Suc 0 ≤ sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2)) dip"
  (is "∀ dip ∈ kD(rt). ?prop dip")
proof
  fix dip
  assume "dip ∈ kD(rt)"
  thus "?prop dip"
  proof (cases "dip = sip")
    assume "dip = sip"
    with ** show ?thesis
      by simp
  next
    assume "dip ≠ sip"
    with ⟨dip ∈ kD(rt)⟩ allkd show ?thesis
      by simp
  qed
qed
} note solve_update [simp] = this

{ fix dip rt dests
  assume *: "∀ ip ∈ dom(dests). ip ∈ kD(rt) ∧ sqn rt ip ≤ the (dests ip)"
  and **: "∀ ip ∈ kD(rt). π3(the (rt ip)) = unk ∨ Suc 0 ≤ sqn rt ip"
  have "∀ dip ∈ kD(rt). π3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"
  proof
    fix dip
    assume "dip ∈ kD(rt)"
    with ** have "π3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip" ..
    thus "π3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"
    proof
      assume "π3(the (rt dip)) = unk" thus ?thesis ..
    next
      assume "Suc 0 ≤ sqn rt dip"
      have "Suc 0 ≤ sqn (invalidate rt dests) dip"
      proof (cases "dip ∈ dom(dests)")
        assume "dip ∈ dom(dests)"
        with * have "sqn rt dip ≤ the (dests dip)" by simp
        with ⟨Suc 0 ≤ sqn rt dip⟩ have "Suc 0 ≤ the (dests dip)" by simp
        with ⟨dip ∈ dom(dests)⟩ ⟨dip ∈ kD(rt)⟩ [THEN kD_Some] show ?thesis
          unfolding invalidate_def sqn_def by auto
      next
        assume "dip ∉ dom(dests)"
        with ⟨Suc 0 ≤ sqn rt dip⟩ ⟨dip ∈ kD(rt)⟩ [THEN kD_Some] show ?thesis
          unfolding invalidate_def sqn_def by auto
      qed
    thus ?thesis by (rule disjI2)
  qed
} note solve_invalidate [simp] = this

show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
      onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn
                              [THEN invariant_restrict_inD]]
      onl_invariant_sterms [OF aodv_wf osn_rreq]
      onl_invariant_sterms [OF aodv_wf dsn_rrep]
      simp add: proj3_inv proj2_eq_sqn)
qed

```

Proposition 7.13

lemma rreq\_rrep\_sn\_any\_step\_invariant:

"paodv i  $\models_A$  (recvmmsg rreq\_rrep\_sn  $\rightarrow$ ) onll  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_)$ . anycast rreq\_rrep\_sn a)"

proof -

have sqnf\_kno: "paodv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ .

( $1 \in \{\text{PRreq-:16..PRreq-:18}\} \rightarrow \text{sqnf}(rt \ \xi) (\text{dip } \xi) = \text{kno}$ )"

by (inv\_cterms inv add: onl\_invariant\_sterms\_TT [OF aodv\_wf addpreRT\_welldefined])

```

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms_TT [OF aadv_wf addpreRT_welldefined]
    onl_invariant_sterms [OF aadv_wf sequence_number_one_or_bigger
        [THEN invariant_restrict_inD]]
    onl_invariant_sterms [OF aadv_wf kD_unk_or_atleast_one]
    onl_invariant_sterms_TT [OF aadv_wf sqnf_kno]
    onl_invariant_sterms [OF aadv_wf osn_rreq]
    onl_invariant_sterms [OF aadv_wf dsn_rrep])
(auto simp: proj2_eq_sqn)

```

qed

Proposition 7.14

lemma rreq\_rrep\_fresh\_any\_step\_invariant:

```
"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast (rreq_rrep_fresh (rt  $\xi$ )) a)"
```

proof -

```

have rreq_oip: "paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .
    (l  $\in$  {PRreq-:3, PRreq-:4, PRreq-:15, PRreq-:27}
     $\longrightarrow$  oip  $\xi \in$  kD(rt  $\xi$ )
     $\wedge$  (sqn (rt  $\xi$ ) (oip  $\xi$ ) > (osn  $\xi$ )
     $\vee$  (sqn (rt  $\xi$ ) (oip  $\xi$ ) = (osn  $\xi$ )
     $\wedge$  the (dhops (rt  $\xi$ ) (oip  $\xi$ ))  $\leq$  Suc (hops  $\xi$ )
     $\wedge$  the (flag (rt  $\xi$ ) (oip  $\xi$ ) = val))))"

```

proof inv\_cterms

```
fix l  $\xi$  l' pp p'
```

```
assume "( $\xi, pp$ )  $\in$  reachable (paadv i) TT"
```

```
and "{PRreq-:2}  $\llbracket \lambda \xi. \xi$  (rt :=
    update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi, kno, val, Suc$  (hops  $\xi$ ), sip  $\xi, \{ \})$   $\rrbracket$  p'  $\in$  sterms  $\Gamma_{AODV}$  pp"
and "l' = PRreq-:3"
```

```

show "osn  $\xi$  < sqn (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi, kno, val, Suc$  (hops  $\xi$ ), sip  $\xi, \{ \}))$  (oip  $\xi$ )
 $\vee$  (sqn (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi, kno, val, Suc$  (hops  $\xi$ ), sip  $\xi, \{ \}))$  (oip  $\xi$ ) = osn  $\xi$ 
 $\wedge$  the (dhops (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi, kno, val, Suc$  (hops  $\xi$ ), sip  $\xi, \{ \}))$  (oip  $\xi$ ))
     $\leq$  Suc (hops  $\xi$ )
 $\wedge$  the (flag (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi, kno, val, Suc$  (hops  $\xi$ ), sip  $\xi, \{ \}))$  (oip  $\xi$ ))
    = val)"

```

```

unfolding update_def by (clarsimp split: option.split)
(metis linorder_neqE_nat not_less)

```

qed

```

have rrep_prrep: "paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .
    (l  $\in$  {PRrep-:2..PRrep-:7}  $\longrightarrow$  (dip  $\xi \in$  kD(rt  $\xi$ )
     $\wedge$  sqn (rt  $\xi$ ) (dip  $\xi$ ) = dsn  $\xi$ 
     $\wedge$  the (dhops (rt  $\xi$ ) (dip  $\xi$ )) = Suc (hops  $\xi$ )
     $\wedge$  the (flag (rt  $\xi$ ) (dip  $\xi$ )) = val
     $\wedge$  the (nhop (rt  $\xi$ ) (dip  $\xi$ ))  $\in$  kD (rt  $\xi$ ))))"

```

```

by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf rrep_1_update_changes]
    onl_invariant_sterms [OF aadv_wf sip_in_kD])

```

show ?thesis

```

by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf rreq_oip]
    onl_invariant_sterms [OF aadv_wf rreq_dip_in_vD_dip_eq_ip]
    onl_invariant_sterms [OF aadv_wf rrep_prrep])

```

qed

Proposition 7.15

lemma rerr\_invalid\_any\_step\_invariant:

```
"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast (rerr_invalid (rt  $\xi$ )) a)"
```

proof -

```

have dests_inv: "paadv i  $\models$ 
    onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ . (l  $\in$  {PAadv-:15, PPkt-:7, PRreq-:9,
    PRreq-:21, PRrep-:10, PRerr-:1}
     $\longrightarrow$  ( $\forall ip \in$  dom(dests  $\xi$ ). ip  $\in$  vD(rt  $\xi$ )))
 $\wedge$  (l  $\in$  {PAadv-:16..PAadv-:19}
     $\cup$  {PPkt-:8..PPkt-:11}
     $\cup$  {PRreq-:10..PRreq-:13})

```

```

      ∪ {PRreq-:22..PRreq-:25}
      ∪ {PRrep-:11..PRrep-:14}
      ∪ {PRerr-:2..PRerr-:5} → (∀ ip ∈ dom(dests ξ). ip ∈ iD(rt ξ)
                                ∧ the (dests ξ ip) = sqn (rt ξ) ip))
      ∧ (l = PPkt-:14 → dip ξ ∈ iD(rt ξ)))"
  by inv_cterms (clarsimp split: split_if_asm option.split_asm simp: domIff)+
show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf dests_inv])
qed

```

Proposition 7.16

Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs  $(rip, rsn)$  is assigned to the variable  $dests$  of type  $ip \rightarrow sqn$ , or to the first argument of the function  $rerr$ , this set is a partial function, i.e., there is at most one entry  $(rip, rsn)$  for each destination  $rip$ : guaranteed by type.

lemma dests\_vD\_inc\_sqn:

```

"paadv i ⊨
  onl ΓAODV (λ(ξ, l). (l ∈ {PAadv-:15, PPkt-:7, PRreq-:9, PRreq-:21, PRrep-:10}
    → (∀ ip ∈ dom(dests ξ). ip ∈ vD(rt ξ) ∧ the (dests ξ ip) = inc (sqn (rt ξ) ip)))
    ∧ (l = PRerr-:1
      → (∀ ip ∈ dom(dests ξ). ip ∈ vD(rt ξ) ∧ the (dests ξ ip) > sqn (rt ξ) ip)))"
  by inv_cterms (clarsimp split: split_if_asm option.split_asm)+

```

Proposition 7.27

lemma route\_tables\_fresher:

```

"paadv i ⊨A (recvmg rreq_rrep_sn →) onll ΓAODV (λ((ξ, _), _, (ξ', _)).
  ∀ dip ∈ kD(rt ξ). rt ξ ⊆dip rt ξ')"

```

proof (inv\_cterms inv add:

```

  onl_invariant_sterms [OF aadv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aadv_wf hop_count_positive [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aadv_wf osn_rreq]
  onl_invariant_sterms [OF aadv_wf dsn_rrep]
  onl_invariant_sterms [OF aadv_wf addpreRT_welldefined [THEN invariant_restrict_inD]])

```

fix ξ pp p'

assume "(ξ, pp) ∈ reachable (paadv i) (recvmg rreq\_rrep\_sn)"

and "{PRreq-:2} [λξ. ξ (|rt := update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {}))] p' ∈ sterms Γ<sub>AODV</sub> pp"

and "Suc 0 ≤ osn ξ"

and \*: "∀ ip ∈ kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) ip)"

show "∀ ip ∈ kD (rt ξ). rt ξ ⊆<sub>ip</sub> update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})"

proof

fix ip

assume "ip ∈ kD (rt ξ)"

moreover with \* have "1 ≤ the (dhops (rt ξ) ip)" by simp

moreover from (Suc 0 ≤ osn ξ)

have "update\_arg\_wf (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})" ..

ultimately show "rt ξ ⊆<sub>ip</sub> update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})"

by (rule rt\_fresher\_update)

qed

next

fix ξ pp p'

assume "(ξ, pp) ∈ reachable (paadv i) (recvmg rreq\_rrep\_sn)"

and "{PRrep-:1} [λξ. ξ (|rt := update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {}))] p' ∈ sterms Γ<sub>AODV</sub> pp"

and "Suc 0 ≤ dsn ξ"

and \*: "∀ ip ∈ kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) ip)"



```

show "∀ip∈kD (rt ξ). rt ξ ⊆ip update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {})"
proof
  fix ip
  assume "ip∈kD (rt ξ)"
  moreover with * have "1 ≤ the (dhops (rt ξ) ip)" by simp
  moreover from ⟨Suc 0 ≤ dsn ξ⟩
    have "update_arg_wf (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {})" ..
  ultimately show "rt ξ ⊆ip update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {})"
    by (rule rt_fresher_update)
qed
end

```

## 0.8 The quality increases predicate

```

theory Quality_Increases
imports Aodv_Predicates Fresher
begin

```

```

definition quality_increases :: "state ⇒ state ⇒ bool"
where "quality_increases ξ ξ' ≡ (∀dip∈kD(rt ξ). dip ∈ kD(rt ξ') ∧ rt ξ ⊆dip rt ξ')
      ∧ (∀dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip)"

```

```

lemma quality_increasesI [intro!]:
  assumes "∧dip. dip ∈ kD(rt ξ) ⇒ dip ∈ kD(rt ξ'"
    and "∧dip. [ dip ∈ kD(rt ξ); dip ∈ kD(rt ξ') ] ⇒ rt ξ ⊆dip rt ξ'"
    and "∧dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip"
  shows "quality_increases ξ ξ'"
  unfolding quality_increases_def using assms by clarsimp

```

```

lemma quality_increasesE [elim]:
  fixes dip
  assumes "quality_increases ξ ξ'"
    and "dip∈kD(rt ξ)"
    and "[ dip ∈ kD(rt ξ'); rt ξ ⊆dip rt ξ'; sqn (rt ξ) dip ≤ sqn (rt ξ') dip ] ⇒ R dip ξ ξ'"
  shows "R dip ξ ξ'"
  using assms unfolding quality_increases_def by clarsimp

```

```

lemma quality_increases_rt_fresherD [dest]:
  fixes ip
  assumes "quality_increases ξ ξ'"
    and "ip∈kD(rt ξ)"
  shows "rt ξ ⊆ip rt ξ'"
  using assms by auto

```

```

lemma quality_increases_sqnE [elim]:
  fixes dip
  assumes "quality_increases ξ ξ'"
    and "sqn (rt ξ) dip ≤ sqn (rt ξ') dip ⇒ R dip ξ ξ'"
  shows "R dip ξ ξ'"
  using assms unfolding quality_increases_def by clarsimp

```

```

lemma quality_increases_refl [intro, simp]: "quality_increases ξ ξ"
  by rule simp_all

```

```

lemma strictly_fresher_quality_increases_right [elim]:
  fixes σ σ' dip
  assumes "rt (σ i) ⊆dip rt (σ nhip)"
    and qinc: "quality_increases (σ nhip) (σ' nhip)"
    and "dip∈kD(rt (σ nhip))"
  shows "rt (σ i) ⊆dip rt (σ' nhip)"
  proof -
    from qinc have "rt (σ nhip) ⊆dip rt (σ' nhip)" using ⟨dip∈kD(rt (σ nhip))⟩

```

```

    by auto
    with ⟨rt (σ i) ⊆dip rt (σ nhip)⟩ show ?thesis ..
qed

lemma kD_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
  shows "i ∈ kD(rt ξ')"
  using assms by auto

lemma kD_nsqn_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
  shows "i ∈ kD(rt ξ') ∧ nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
  proof -
    from assms have "i ∈ kD(rt ξ')" ..
    moreover with assms have "rt ξ ⊆i rt ξ'" by auto
    ultimately have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
      using ⟨i ∈ kD(rt ξ)⟩ by - (erule(2) rt_fresher_imp_nsqn_le)
    with ⟨i ∈ kD(rt ξ')⟩ show ?thesis ..
  qed

lemma nsqn_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
  shows "nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
  using assms by (rule kD_nsqn_quality_increases [THEN conjunct2])

lemma kD_nsqn_quality_increases_trans [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "s ≤ nsqn (rt ξ) i"
    and "quality_increases ξ ξ'"
  shows "i ∈ kD(rt ξ') ∧ s ≤ nsqn (rt ξ') i"
  proof
    from ⟨i ∈ kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ show "i ∈ kD(rt ξ')" ..
  next
    from ⟨i ∈ kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
    with ⟨s ≤ nsqn (rt ξ) i⟩ show "s ≤ nsqn (rt ξ') i" by (rule le_trans)
  qed

lemma nsqn_quality_increases_nsqn_lt_lt [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "s < nsqn (rt ξ) i"
  shows "s < nsqn (rt ξ') i"
  proof -
    from assms(1-2) have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
    with ⟨s < nsqn (rt ξ) i⟩ show "s < nsqn (rt ξ') i" by simp
  qed

lemma nsqn_quality_increases_dhops [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "nsqn (rt ξ) i = nsqn (rt ξ') i"
  shows "the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i)"
  using assms unfolding quality_increases_def
  by (clarsimp) (drule(1) bspec, clarsimp simp: rt_fresher_def2)

lemma nsqn_quality_increases_nsqn_eq_le [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "s = nsqn (rt ξ) i"
  shows "s < nsqn (rt ξ') i ∨ (s = nsqn (rt ξ') i ∧ the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i))"
  using assms by (metis nat_less_le nsqn_quality_increases nsqn_quality_increases_dhops)

```

```

lemma quality_increases_rreq_rrep_props [elim]:
  fixes sn ip hops sip
  assumes qinc: "quality_increases (σ sip) (σ' sip)"
    and "1 ≤ sn"
    and *: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
           ∧ (nsqn (rt (σ sip)) ip = sn
              → (the (dhops (rt (σ sip)) ip) ≤ hops
                  ∨ the (flag (rt (σ sip)) ip) = inv))"
  shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip
        ∧ (nsqn (rt (σ' sip)) ip = sn
           → (the (dhops (rt (σ' sip)) ip) ≤ hops
               ∨ the (flag (rt (σ' sip)) ip) = inv))"

  (is "_ ∧ ?nsqnafter")
proof -
  from * obtain "ip ∈ kD(rt (σ sip))" and "sn ≤ nsqn (rt (σ sip)) ip" by auto

  from ⟨quality_increases (σ sip) (σ' sip)⟩
    have "sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip" ..
  from ⟨quality_increases (σ sip) (σ' sip)⟩ and ⟨ip ∈ kD (rt (σ sip))⟩
    have "ip ∈ kD (rt (σ' sip))" ..

  from ⟨sn ≤ nsqn (rt (σ sip)) ip⟩ have ?nsqnafter
proof
  assume "sn < nsqn (rt (σ sip)) ip"
  also from ⟨ip ∈ kD (rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
    have "... ≤ nsqn (rt (σ' sip)) ip" ..
  finally have "sn < nsqn (rt (σ' sip)) ip" .
  thus ?thesis by simp
next
  assume "sn = nsqn (rt (σ sip)) ip"
  with ⟨ip ∈ kD (rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
    have "sn < nsqn (rt (σ' sip)) ip
          ∨ (sn = nsqn (rt (σ' sip)) ip
              ∧ the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip))" ..
  hence "sn < nsqn (rt (σ' sip)) ip
        ∨ (nsqn (rt (σ' sip)) ip = sn ∧ (the (dhops (rt (σ' sip)) ip) ≤ hops
                                             ∨ the (flag (rt (σ' sip)) ip) = inv))"
proof
  assume "sn < nsqn (rt (σ' sip)) ip" thus ?thesis ..
next
  assume "sn = nsqn (rt (σ' sip)) ip
        ∧ the (dhops (rt (σ sip)) ip) ≥ the (dhops (rt (σ' sip)) ip)"
  hence "sn = nsqn (rt (σ' sip)) ip"
    and "the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)" by auto

  from * and ⟨sn = nsqn (rt (σ sip)) ip⟩ have "the (dhops (rt (σ sip)) ip) ≤ hops
        ∨ the (flag (rt (σ sip)) ip) = inv"
  by simp
  thus ?thesis
proof
  assume "the (dhops (rt (σ sip)) ip) ≤ hops"
  with ⟨the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)⟩
    have "the (dhops (rt (σ' sip)) ip) ≤ hops" by simp
  with ⟨sn = nsqn (rt (σ' sip)) ip⟩ show ?thesis by simp
next
  assume "the (flag (rt (σ sip)) ip) = inv"
  with ⟨ip ∈ kD (rt (σ sip))⟩ have "nsqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip - 1" ..

  with ⟨sn ≥ 1⟩ and ⟨sn = nsqn (rt (σ sip)) ip⟩
    have "sqn (rt (σ sip)) ip > 1" by simp

  from ⟨ip ∈ kD (rt (σ' sip))⟩ show ?thesis
proof (rule vD_or_iD)

```

```

    assume "ip∈iD(rt (σ' sip))"
    hence "the (flag (rt (σ' sip)) ip) = inv" ..
    with (sn = nsqn (rt (σ' sip)) ip) show ?thesis
      by simp
  next

```

```

    assume "ip∈vD(rt (σ' sip))"
    hence "nsqn (rt (σ' sip)) ip = sqn (rt (σ' sip)) ip" ..
    with (sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip)
      have "nsqn (rt (σ' sip)) ip ≥ sqn (rt (σ sip)) ip" by simp

```

```

    with (sqn (rt (σ sip)) ip > 1)
      have "nsqn (rt (σ' sip)) ip > sqn (rt (σ sip)) ip - 1" by simp
    with (nsqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip - 1)
      have "nsqn (rt (σ' sip)) ip > nsqn (rt (σ sip)) ip" by simp
    with (sn = nsqn (rt (σ sip)) ip) have "nsqn (rt (σ' sip)) ip > sn"
      by simp
    thus ?thesis ..

```

qed

qed

qed

thus ?thesis by (metis (mono\_tags) le\_cases not\_le)

qed

with (ip∈kD (rt (σ' sip))) show "ip∈kD (rt (σ' sip)) ∧ ?nsqnafter" ..

qed

lemma quality\_increases\_rreq\_rrep\_props':

fixes sn ip hops sip

assumes "∀j. quality\_increases (σ j) (σ' j)"

and "1 ≤ sn"

and \*: "ip∈kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip  
 ∧ (nsqn (rt (σ sip)) ip = sn  
 → (the (dhops (rt (σ sip)) ip) ≤ hops  
 ∨ the (flag (rt (σ sip)) ip) = inv))"

shows "ip∈kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip  
 ∧ (nsqn (rt (σ' sip)) ip = sn  
 → (the (dhops (rt (σ' sip)) ip) ≤ hops  
 ∨ the (flag (rt (σ' sip)) ip) = inv))"

proof -

from assms(1) have "quality\_increases (σ sip) (σ' sip)" ..

thus ?thesis using assms(2-3) by (rule quality\_increases\_rreq\_rrep\_props)

qed

lemma rteq\_quality\_increases:

assumes "∀j. j ≠ i → quality\_increases (σ j) (σ' j)"

and "rt (σ' i) = rt (σ i)"

shows "∀j. quality\_increases (σ j) (σ' j)"

using assms by clarsimp (metis order\_refl quality\_increasesI rt\_fresher\_refl)

definition msg\_fresh :: "(ip ⇒ state) ⇒ msg ⇒ bool"

where "msg\_fresh σ m ≡

```

  case m of Rreq hopsc _ _ _ oipc osnc sipc ⇒ osnc ≥ 1 ∧ (sipc ≠ oipc →
    oipc∈kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) oipc ≥ osnc
    ∧ (nsqn (rt (σ sipc)) oipc = osnc
      → (hopsc ≥ the (dhops (rt (σ sipc)) oipc)
        ∨ the (flag (rt (σ sipc)) oipc) = inv)))
  | Rrep hopsc dipc dsnc _ sipc ⇒ dsnc ≥ 1 ∧ (sipc ≠ dipc →
    dipc∈kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) dipc ≥ dsnc
    ∧ (nsqn (rt (σ sipc)) dipc = dsnc
      → (hopsc ≥ the (dhops (rt (σ sipc)) dipc)
        ∨ the (flag (rt (σ sipc)) dipc) = inv)))
  | Rerr destsc sipc ⇒ (∀ripc∈dom(destsc). (ripc∈kD(rt (σ sipc))
    ∧ the (destsc ripc) - 1 ≤ nsqn (rt (σ sipc)) ripc))
  | _ ⇒ True"

```

lemma msg\_fresh [simp]:

```
"^hops rreqid dip dsn dsk oip osn sip.
  msg_fresh  $\sigma$  (Rreq hops rreqid dip dsn dsk oip osn sip) =
    (osn  $\geq$  1  $\wedge$  (sip  $\neq$  oip  $\longrightarrow$  oip $\in$ kD(rt ( $\sigma$  sip))
       $\wedge$  nsqn (rt ( $\sigma$  sip)) oip  $\geq$  osn
       $\wedge$  (nsqn (rt ( $\sigma$  sip)) oip = osn
         $\longrightarrow$  (hops  $\geq$  the (dhops (rt ( $\sigma$  sip)) oip)
           $\vee$  the (flag (rt ( $\sigma$  sip)) oip) = inv))))"

"^hops dip dsn oip sip. msg_fresh  $\sigma$  (Rrep hops dip dsn oip sip) =
  (dsn  $\geq$  1  $\wedge$  (sip  $\neq$  dip  $\longrightarrow$  dip $\in$ kD(rt ( $\sigma$  sip))
     $\wedge$  nsqn (rt ( $\sigma$  sip)) dip  $\geq$  dsn
     $\wedge$  (nsqn (rt ( $\sigma$  sip)) dip = dsn
       $\longrightarrow$  (hops  $\geq$  the (dhops (rt ( $\sigma$  sip)) dip))
         $\vee$  the (flag (rt ( $\sigma$  sip)) dip) = inv))))"

"^dests sip.      msg_fresh  $\sigma$  (Rerr dests sip) =
  ( $\forall$  ripc $\in$ dom(dests). (ripc $\in$ kD(rt ( $\sigma$  sip))
     $\wedge$  the (dests ripc) - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) ripc))"

"^d dip.          msg_fresh  $\sigma$  (Newpkt d dip) = True"
"^d dip sip.      msg_fresh  $\sigma$  (Pkt d dip sip) = True"
unfolding msg_fresh_def by simp_all
```

lemma msg\_fresh\_inc\_sn [simp, elim]:

```
"msg_fresh  $\sigma$  m  $\implies$  rreq_rrep_sn m"
by (cases m) simp_all
```

lemma recv\_msg\_fresh\_inc\_sn [simp, elim]:

```
"orecvmsg (msg_fresh)  $\sigma$  m  $\implies$  recvmmsg rreq_rrep_sn m"
by (cases m) simp_all
```

lemma rreq\_nsqn\_is\_fresh [simp]:

```
fixes  $\sigma$  msg hops rreqid dip dsn dsk oip osn sip
assumes "rreq_rrep_fresh (rt ( $\sigma$  sip)) (Rreq hops rreqid dip dsn dsk oip osn sip)"
  and "rreq_rrep_sn (Rreq hops rreqid dip dsn dsk oip osn sip)"
shows "msg_fresh  $\sigma$  (Rreq hops rreqid dip dsn dsk oip osn sip)"
  (is "msg_fresh  $\sigma$  ?msg")
```

proof -

```
let ?rt = "rt ( $\sigma$  sip)"
from assms(2) have "1  $\leq$  osn" by simp
thus ?thesis
unfolding msg_fresh_def
proof (simp only: msg.case, intro conjI impI)
  assume "sip  $\neq$  oip"
  with assms(1) show "oip  $\in$  kD(?rt)" by simp
next
  assume "sip  $\neq$  oip"
  and "nsqn ?rt oip = osn"
  show "the (dhops ?rt oip)  $\leq$  hops  $\vee$  the (flag ?rt oip) = inv"
  proof (cases "oip $\in$ vD(?rt)")
    assume "oip $\in$ vD(?rt)"
    hence "nsqn ?rt oip = sqn ?rt oip" ..
    with (nsqn ?rt oip = osn) have "sqn ?rt oip = osn" by simp
    with assms(1) and (sip  $\neq$  oip) have "the (dhops ?rt oip)  $\leq$  hops"
      by simp
    thus ?thesis ..
  next
    assume "oip $\notin$ vD(?rt)"
    moreover from assms(1) and (sip  $\neq$  oip) have "oip $\in$ kD(?rt)" by simp
    ultimately have "oip $\in$ iD(?rt)" by auto
    hence "the (flag ?rt oip) = inv" ..
    thus ?thesis ..
  qed
next
  assume "sip  $\neq$  oip"
```

```

with assms(1) have "osn ≤ sqn ?rt oip" by auto
thus "osn ≤ nsqn (rt (σ sip)) oip"
proof (rule nat_le_eq_or_lt)
  assume "osn < sqn ?rt oip"
  hence "osn ≤ sqn ?rt oip - 1" by simp
  also have "... ≤ nsqn ?rt oip" by (rule sqn_nsqn)
  finally show "osn ≤ nsqn ?rt oip" .
next
assume "osn = sqn ?rt oip"
with assms(1) and ⟨sip ≠ oip⟩ have "oip ∈ kD(?rt)"
  and "the (flag ?rt oip) = val"

  by auto
  hence "nsqn ?rt oip = sqn ?rt oip" ..
  with ⟨osn = sqn ?rt oip⟩ have "nsqn ?rt oip = osn" by simp
  thus "osn ≤ nsqn ?rt oip" by simp
qed
qed simp
qed

lemma rrep_nsqn_is_fresh [simp]:
  fixes σ msg hops dip dsn oip sip
  assumes "rreq_rrep_fresh (rt (σ sip)) (Rrep hops dip dsn oip sip)"
  and "rreq_rrep_sn (Rrep hops dip dsn oip sip)"
  shows "msg_fresh σ (Rrep hops dip dsn oip sip)"
  (is "msg_fresh σ ?msg")
proof -
  let ?rt = "rt (σ sip)"
  from assms have "sip ≠ dip → dip ∈ kD(?rt) ∧ sqn ?rt dip = dsn ∧ the (flag ?rt dip) = val"
  by simp
  hence "sip ≠ dip → dip ∈ kD(?rt) ∧ nsqn ?rt dip ≥ dsn"
  by clarsimp
  with assms show "msg_fresh σ ?msg"
  by clarsimp
qed

lemma rerr_nsqn_is_fresh [simp]:
  fixes σ msg dests sip
  assumes "rerr_invalid (rt (σ sip)) (Rerr dests sip)"
  shows "msg_fresh σ (Rerr dests sip)"
  (is "msg_fresh σ ?msg")
proof -
  let ?rt = "rt (σ sip)"
  from assms have *: "(∀ rip ∈ dom(dests). (rip ∈ iD(rt (σ sip))
    ∧ the (dests rip) = sqn (rt (σ sip)) rip))"
  by clarsimp
  have "(∀ rip ∈ dom(dests). (rip ∈ kD(rt (σ sip))
    ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip))"
  proof
    fix rip
    assume "rip ∈ dom dests"
    with * have "rip ∈ iD(rt (σ sip))" and "the (dests rip) = sqn (rt (σ sip)) rip"
    by auto

    from this(2) have "the (dests rip) - 1 = sqn (rt (σ sip)) rip - 1" by simp
    also have "... ≤ nsqn (rt (σ sip)) rip" by (rule sqn_nsqn)
    finally have "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip" .

    with ⟨rip ∈ iD(rt (σ sip))⟩
    show "rip ∈ kD(rt (σ sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
    by clarsimp
  qed
  thus "msg_fresh σ ?msg"
  by simp
qed

```

```

lemma quality_increases_msg_fresh [elim]:
  assumes qinc: " $\forall j. \text{quality\_increases } (\sigma \ j) (\sigma' \ j)$ "
    and "msg_fresh  $\sigma \ m$ "
  shows "msg_fresh  $\sigma' \ m$ "
using assms(2)
proof (cases m)
  fix hops rreqid dip dsn dsk oip osn sip
  assume [simp]: " $m = Rreq \ hops \ rreqid \ dip \ dsn \ dsk \ oip \ osn \ sip$ "
    and "msg_fresh  $\sigma \ m$ "
  then have " $osn \geq 1$ " and " $sip = oip \vee (oip \in kD(rt \ (\sigma \ sip)) \wedge osn \leq nsqn \ (rt \ (\sigma \ sip)) \ oip$ 
     $\wedge (nsqn \ (rt \ (\sigma \ sip)) \ oip = osn$ 
     $\longrightarrow (the \ (dhops \ (rt \ (\sigma \ sip)) \ oip) \leq hops$ 
     $\vee the \ (flag \ (rt \ (\sigma \ sip)) \ oip) = inv))$ "

    by auto
  from this(2) show ?thesis
proof
  assume "sip = oip" with  $\langle osn \geq 1 \rangle$  show ?thesis by simp
next
  assume " $oip \in kD(rt \ (\sigma \ sip)) \wedge osn \leq nsqn \ (rt \ (\sigma \ sip)) \ oip$ 
     $\wedge (nsqn \ (rt \ (\sigma \ sip)) \ oip = osn$ 
     $\longrightarrow (the \ (dhops \ (rt \ (\sigma \ sip)) \ oip) \leq hops$ 
     $\vee the \ (flag \ (rt \ (\sigma \ sip)) \ oip) = inv))$ "
  moreover from qinc have "quality_increases  $(\sigma \ sip) (\sigma' \ sip)$ " ..
  ultimately have " $oip \in kD(rt \ (\sigma' \ sip)) \wedge osn \leq nsqn \ (rt \ (\sigma' \ sip)) \ oip$ 
     $\wedge (nsqn \ (rt \ (\sigma' \ sip)) \ oip = osn$ 
     $\longrightarrow (the \ (dhops \ (rt \ (\sigma' \ sip)) \ oip) \leq hops$ 
     $\vee the \ (flag \ (rt \ (\sigma' \ sip)) \ oip) = inv))$ "

    using  $\langle osn \geq 1 \rangle$  by (rule quality_increases_rreq_rrep_props [rotated 2])
  with  $\langle osn \geq 1 \rangle$  show "msg_fresh  $\sigma' \ m$ "
    by (clarsimp)
qed
next
  fix hops dip dsn oip sip
  assume [simp]: " $m = Rrep \ hops \ dip \ dsn \ oip \ sip$ "
    and "msg_fresh  $\sigma \ m$ "
  then have " $dsn \geq 1$ " and " $sip = dip \vee (dip \in kD(rt \ (\sigma \ sip)) \wedge dsn \leq nsqn \ (rt \ (\sigma \ sip)) \ dip$ 
     $\wedge (nsqn \ (rt \ (\sigma \ sip)) \ dip = dsn$ 
     $\longrightarrow (the \ (dhops \ (rt \ (\sigma \ sip)) \ dip) \leq hops$ 
     $\vee the \ (flag \ (rt \ (\sigma \ sip)) \ dip) = inv))$ "

    by auto
  from this(2) show "?thesis"
proof
  assume "sip = dip" with  $\langle dsn \geq 1 \rangle$  show ?thesis by simp
next
  assume " $dip \in kD(rt \ (\sigma \ sip)) \wedge dsn \leq nsqn \ (rt \ (\sigma \ sip)) \ dip$ 
     $\wedge (nsqn \ (rt \ (\sigma \ sip)) \ dip = dsn$ 
     $\longrightarrow (the \ (dhops \ (rt \ (\sigma \ sip)) \ dip) \leq hops$ 
     $\vee the \ (flag \ (rt \ (\sigma \ sip)) \ dip) = inv))$ "
  moreover from qinc have "quality_increases  $(\sigma \ sip) (\sigma' \ sip)$ " ..
  ultimately have " $dip \in kD(rt \ (\sigma' \ sip)) \wedge dsn \leq nsqn \ (rt \ (\sigma' \ sip)) \ dip$ 
     $\wedge (nsqn \ (rt \ (\sigma' \ sip)) \ dip = dsn$ 
     $\longrightarrow (the \ (dhops \ (rt \ (\sigma' \ sip)) \ dip) \leq hops$ 
     $\vee the \ (flag \ (rt \ (\sigma' \ sip)) \ dip) = inv))$ "

    using  $\langle dsn \geq 1 \rangle$  by (rule quality_increases_rreq_rrep_props [rotated 2])
  with  $\langle dsn \geq 1 \rangle$  show "msg_fresh  $\sigma' \ m$ "
    by (clarsimp)
qed
next
  fix dests sip
  assume [simp]: " $m = Rerr \ dests \ sip$ "
    and "msg_fresh  $\sigma \ m$ "
  then have *: " $\forall rip \in dom(dests). rip \in kD(rt \ (\sigma \ sip))$ 
     $\wedge the \ (dests \ rip) - 1 \leq nsqn \ (rt \ (\sigma \ sip)) \ rip$ "

```

```

by simp
have "∀rip∈dom(dests). rip∈kD(rt (σ' sip))
      ∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip"
proof
  fix rip
  assume "rip∈dom(dests)"
  with * have "rip∈kD(rt (σ sip))" and "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
    by - (drule(1) bspec, clarsimp)+
  moreover from qinc have "quality_increases (σ sip) (σ' sip)" by simp
  ultimately show "rip∈kD(rt (σ' sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip" ..
qed
thus ?thesis by simp
qed simp_all
end

```

## 0.9 The ‘open’ AODV model

```

theory OAodv
imports Aodv "../AWN/OAWN_SOS_Labels" "../AWN/OAWN_Convert"
begin

```

Definitions for stating and proving global network properties over individual processes.

```

definition σAODV' :: "(ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp) set"
where "σAODV' ≡ {(λi. aodv_init i, ΓAODV PAodv)}"

```

abbreviation opaodv

```

:: "ip ⇒ ((ip ⇒ state) × (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
where
  "opaodv i ≡ (| init = σAODV', trans = oseqp_sos ΓAODV i |)"

```

```

lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))"
  unfolding σAODV_def σAODV'_def by rule simp_all

```

```

lemma oaodv_control_within [simp]: "control_within ΓAODV (init (opaodv i))"
  unfolding σAODV'_def by (rule control_withinI) (auto simp del: ΓAODV_simps)

```

```

lemma σAODV'_labels [simp]: "(σ, p) ∈ σAODV' ⇒ labels ΓAODV p = {PAodv-.:0}"
  unfolding σAODV'_def by simp

```

```

lemma oaodv_init_kD_empty [simp]:
  "(σ, p) ∈ σAODV' ⇒ kD (rt (σ i)) = {}"
  unfolding σAODV'_def kD_def by simp

```

```

lemma oaodv_init_vD_empty [simp]:
  "(σ, p) ∈ σAODV' ⇒ vD (rt (σ i)) = {}"
  unfolding σAODV'_def vD_def by simp

```

```

lemma oaodv_trans: "trans (opaodv i) = oseqp_sos ΓAODV i"
  by simp

```

declare

```

oseq_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]
oseq_step_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]

```

end

## 0.10 Global invariant proofs over sequential processes

```

theory Global_Invariants
imports Seq_Invariants
  Aodv_Predicates
  Fresher

```



```

    Quality_Increases
    "../AWN/OAWN_Convert"
    "OAadv"
begin
lemma other_quality_increases [elim]:
  assumes "other quality_increases I  $\sigma$   $\sigma'$ "
  shows " $\forall j$ . quality_increases ( $\sigma$  j) ( $\sigma'$  j)"
  using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
  fixes m
  assumes *: "otherwith P I (orecvmsg Q)  $\sigma$   $\sigma'$  a"
  and weakenP: " $\bigwedge \sigma m$ . P  $\sigma$  m  $\implies$  P'  $\sigma$  m"
  and weakenQ: " $\bigwedge \sigma m$ . Q  $\sigma$  m  $\implies$  Q'  $\sigma$  m"
  shows "otherwith P' I (orecvmsg Q')  $\sigma$   $\sigma'$  a"
proof
  fix j
  assume "j  $\notin$  I"
  with * have "P ( $\sigma$  j) ( $\sigma'$  j)" by auto
  thus "P' ( $\sigma$  j) ( $\sigma'$  j)" by (rule weakenP)
next
  from * have "orecvmsg Q  $\sigma$  a" by auto
  thus "orecvmsg Q'  $\sigma$  a"
  by rule (erule weakenQ)
qed

lemma oreceived_msg_inv:
  assumes other: " $\bigwedge \sigma \sigma' m$ .  $\llbracket$  P  $\sigma$  m; other Q {i}  $\sigma$   $\sigma'$   $\rrbracket \implies$  P  $\sigma'$  m"
  and local: " $\bigwedge \sigma m$ . P  $\sigma$  m  $\implies$  P ( $\sigma$ (i :=  $\sigma$  i(msg := m))) m"
  shows "opaadv i  $\models$  (otherwith Q {i} (orecvmsg P), other Q {i}  $\rightarrow$ )
    onl  $\Gamma_{AODV}$  ( $\lambda(\sigma, l)$ . l  $\in$  {PAadv-:1}  $\longrightarrow$  P  $\sigma$  (msg ( $\sigma$  i)))"
proof (inv_cterms, intro impI)
  fix  $\sigma \sigma' l$ 
  assume "l = PAadv-:1  $\longrightarrow$  P  $\sigma$  (msg ( $\sigma$  i))"
  and "l = PAadv-:1"
  and "other Q {i}  $\sigma \sigma'$ "
  from this(1-2) have "P  $\sigma$  (msg ( $\sigma$  i))" ..
  hence "P  $\sigma'$  (msg ( $\sigma$  i))" using (other Q {i}  $\sigma \sigma'$ )
  by (rule other)
  moreover from (other Q {i}  $\sigma \sigma'$ ) have " $\sigma' i = \sigma i$ " ..
  ultimately show "P  $\sigma'$  (msg ( $\sigma' i$ ))" by simp
next
  fix  $\sigma \sigma' msg$ 
  assume "otherwith Q {i} (orecvmsg P)  $\sigma \sigma'$  (receive msg)"
  and " $\sigma' i = \sigma i$ (msg := msg)"
  from this(1) have "P  $\sigma$  msg"
  and " $\forall j$ . j  $\neq$  i  $\longrightarrow$  Q ( $\sigma$  j) ( $\sigma'$  j)" by auto
  from this(1) have "P ( $\sigma$ (i :=  $\sigma$  i(msg := msg))) msg" by (rule local)
  thus "P  $\sigma'$  msg"
proof (rule other)
  from (  $\sigma' i = \sigma i$ (msg := msg) ) and (  $\forall j$ . j  $\neq$  i  $\longrightarrow$  Q ( $\sigma$  j) ( $\sigma'$  j) )
  show "other Q {i} ( $\sigma$ (i :=  $\sigma$  i(msg := msg)))  $\sigma'$ "
  by - (rule otherI, auto)
qed
qed

```

(Equivalent to) Proposition 7.27

```

lemma local_quality_increases:
  "paadv i  $\models_A$  (recvmsg rreq_rrep_sn  $\rightarrow$ ) onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_))$ . quality_increases  $\xi \xi'$ )"
proof (rule step_invariantI)
  fix s a s'
  assume sr: "s  $\in$  reachable (paadv i) (recvmsg rreq_rrep_sn)"
  and tr: "(s, a, s')  $\in$  trans (paadv i)"

```

```

    and rm: "recvmsg rreq_rrep_sn a"
  from sr have srTT: "s ∈ reachable (paadv i) TT" ..

  from route_tables_fresher sr tr rm
    have "onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \forall dip \in kD (rt \xi). rt \xi \sqsubseteq_{dip} rt \xi') (s, a, s')$ "
      by (rule step_invariantD)

  moreover from known_destinations_increase srTT tr TT_True
    have "onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). kD (rt \xi) \subseteq kD (rt \xi')) (s, a, s')$ "
      by (rule step_invariantD)

  moreover from sqns_increase srTT tr TT_True
    have "onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \forall ip. sqn (rt \xi) ip \leq sqn (rt \xi') ip) (s, a, s')$ "
      by (rule step_invariantD)

  ultimately show "onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). quality\_increases \xi \xi') (s, a, s')$ "
    unfolding onll_def by auto
qed

lemmas olocal_quality_increases =
  open_seq_step_invariant [OF local_quality_increases initiali_aadv oaadv_trans aadv_trans,
    simplified seqll_onll_swap]

lemma oquality_increases:
  "opaadv i  $\models_A$  (otherwith quality_increases {i} (orecvmsg ( $\lambda\_.$  rreq_rrep_sn)),
    other quality_increases {i}  $\rightarrow$ )
    onll  $\Gamma_{AODV} (\lambda((\sigma, \_), \_, (\sigma', \_)). \forall j. quality\_increases (\sigma j) (\sigma' j))"$ 
  (is " $\_ \models_A (?S, \_ \rightarrow) \_$ ")
proof (rule onll_ostep_invariantI, simp)
  fix  $\sigma$  p l a  $\sigma'$  p' l'
  assume or: " $(\sigma, p) \in oreachable (opaadv i) ?S (other\ quality\_increases\ \{i\})"$ "
    and ll: " $l \in labels\ \Gamma_{AODV}\ p"$ "
    and " $?S\ \sigma\ \sigma'\ a$ "
    and tr: " $((\sigma, p), a, (\sigma', p')) \in oseqp\_sos\ \Gamma_{AODV}\ i$ "
    and ll': " $l' \in labels\ \Gamma_{AODV}\ p'$ "
  from this(1-3) have "orecvmsg ( $\lambda\_.$  rreq_rrep_sn)  $\sigma$  a"
    by (auto dest!: oreachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"
      and QU="other quality_increases {i}"]
      otherwith_actionD)
  with or have orw: " $(\sigma, p) \in oreachable (opaadv i) (act (recvmsg\ rreq\_rrep\_sn))$ 
    (other quality_increases {i})"
    by - (erule oreachable_weakenE, auto)
  with tr ll ll' and (orecvmsg ( $\lambda\_.$  rreq_rrep_sn)  $\sigma$  a) have "quality_increases ( $\sigma$  i) ( $\sigma'$  i)"
    by - (drule onll_ostep_invariantD [OF olocal_quality_increases], auto simp: seqll_def)
  with (?S  $\sigma$   $\sigma'$  a) show " $\forall j. quality\_increases (\sigma j) (\sigma' j)$ "
    by (auto dest!: otherwith_syncD)
qed

lemma rreq_rrep_nsqn_fresh_any_step_invariant:
  "opaadv i  $\models_A$  (act (recvmsg rreq_rrep_sn), other A {i}  $\rightarrow$ )
    onll  $\Gamma_{AODV} (\lambda((\sigma, \_), a, \_). anycast (msg\_fresh\ \sigma)\ a)"$ 
proof (rule ostep_invariantI, simp del: act_simp)
  fix  $\sigma$  p a  $\sigma'$  p'
  assume or: " $(\sigma, p) \in oreachable (opaadv i) (act (recvmsg\ rreq\_rrep\_sn)) (other\ A\ \{i\})"$ "
    and " $((\sigma, p), a, (\sigma', p')) \in oseqp\_sos\ \Gamma_{AODV}\ i$ "
    and recv: "act (recvmsg rreq_rrep_sn)  $\sigma$   $\sigma'$  a"
  obtain l l' where "l ∈ labels  $\Gamma_{AODV}$  p" and "l' ∈ labels  $\Gamma_{AODV}$  p'"
    by (metis aadv_ex_label)
  from  $((\sigma, p), a, (\sigma', p')) \in oseqp\_sos\ \Gamma_{AODV}\ i$ 
    have tr: " $((\sigma, p), a, (\sigma', p')) \in trans (opaadv i)$ " by simp
  have "anycast (rreq_rrep_fresh (rt ( $\sigma$  i))) a"
proof -
  have "opaadv i  $\models_A$  (act (recvmsg rreq_rrep_sn), other A {i}  $\rightarrow$ )"

```

```

      onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_)$ . anycast (rreq_rrep_fresh (rt  $\xi$ )) a))"
by (rule ostep_invariant_weakenE [OF
  open_seq_step_invariant [OF rreq_rrep_fresh_any_step_invariant initiali_aadv,
    simplified seqll_onll_swap]]) auto
hence "onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_)$ . anycast (rreq_rrep_fresh (rt  $\xi$ )) a))
  (( $\sigma, p$ ), a, ( $\sigma', p'$ ))"
  using or tr recv by - (erule(4) ostep_invariantE)
thus ?thesis
  using  $\langle l \in \text{labels } \Gamma_{AODV} p \rangle$  and  $\langle l' \in \text{labels } \Gamma_{AODV} p' \rangle$  by auto
qed

moreover have "anycast (rerr_invalid (rt ( $\sigma$  i))) a"
proof -
  have "opaadv i  $\models_A$  (act (recvmmsg rreq_rrep_sn), other A {i}  $\rightarrow$ )
    onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_)$ . anycast (rerr_invalid (rt  $\xi$ )) a))"
  by (rule ostep_invariant_weakenE [OF
    open_seq_step_invariant [OF rerr_invalid_any_step_invariant initiali_aadv,
      simplified seqll_onll_swap]]) auto
  hence "onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_)$ . anycast (rerr_invalid (rt  $\xi$ )) a))
    (( $\sigma, p$ ), a, ( $\sigma', p'$ ))"
    using or tr recv by - (erule(4) ostep_invariantE)
  thus ?thesis
    using  $\langle l \in \text{labels } \Gamma_{AODV} p \rangle$  and  $\langle l' \in \text{labels } \Gamma_{AODV} p' \rangle$  by auto
  qed

moreover have "anycast rreq_rrep_sn a"
proof -
  from or tr recv
  have "onll  $\Gamma_{AODV}$  (seqll i ( $\lambda(\_, a, \_)$ . anycast rreq_rrep_sn a)) (( $\sigma, p$ ), a, ( $\sigma', p'$ ))"
  by (rule ostep_invariantE [OF
    open_seq_step_invariant [OF rreq_rrep_sn_any_step_invariant initiali_aadv
      oaadv_trans aadv_trans,
      simplified seqll_onll_swap]])
  thus ?thesis
    using  $\langle l \in \text{labels } \Gamma_{AODV} p \rangle$  and  $\langle l' \in \text{labels } \Gamma_{AODV} p' \rangle$  by auto
  qed

moreover have "anycast ( $\lambda m$ . not_Pkt m  $\rightarrow$  msg_sender m = i) a"
proof -
  have "opaadv i  $\models_A$  (act (recvmmsg rreq_rrep_sn), other A {i}  $\rightarrow$ )
    onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_)$ . anycast ( $\lambda m$ . not_Pkt m  $\rightarrow$  msg_sender m = i) a))"
  by (rule ostep_invariant_weakenE [OF
    open_seq_step_invariant [OF sender_ip_valid initiali_aadv,
      simplified seqll_onll_swap]]) auto
  thus ?thesis using or tr recv  $\langle l \in \text{labels } \Gamma_{AODV} p \rangle$  and  $\langle l' \in \text{labels } \Gamma_{AODV} p' \rangle$ 
    by - (drule(3) onll_ostep_invariantD, auto)
  qed

ultimately have "anycast (msg_fresh  $\sigma$ ) a"
  by (simp_all add: anycast_def
    del: msg_fresh
    split: seq_action.split_asm msg.split_asm) simp_all
thus "onll  $\Gamma_{AODV}$  ( $\lambda((\sigma, \_), a, \_)$ . anycast (msg_fresh  $\sigma$ ) a) (( $\sigma, p$ ), a, ( $\sigma', p'$ ))"
  by auto
qed

lemma oreceived_rreq_rrep_nsqn_fresh_inv:
"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onll  $\Gamma_{AODV}$  ( $\lambda(\sigma, l)$ .  $l \in \{\text{PAadv}::1\} \rightarrow$  msg_fresh  $\sigma$  (msg ( $\sigma$  i)))"
proof (rule oreceived_msg_inv)
  fix  $\sigma \sigma' m$ 
  assume *: "msg_fresh  $\sigma$  m"
  and "other quality_increases {i}  $\sigma \sigma'$ "

```

```

from this(2) have "∀j. quality_increases (σ j) (σ' j)" ..
thus "msg_fresh σ' m" using * ..
next
  fix σ m
  assume "msg_fresh σ m"
  thus "msg_fresh (σ(i := σ i(|msg := m|))) m"
  proof (cases m)
    fix dests sip
    assume "m = Rerr dests sip"
    with ⟨msg_fresh σ m⟩ show ?thesis
    by auto
  qed auto
qed

lemma oquality_increases_nsqn_fresh:
  "opaadv i ⊨A (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onl ΓAODV (λ((σ, _), _, (σ', _)). ∀j. quality_increases (σ j) (σ' j))"
  by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
  "opaadv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onl ΓAODV (seq1 i (λ(ξ, l). l ∈ {PAadv-:4, PAadv-:5} ∪ {PRreq-:n | n. True} → 1 ≤ osn ξ))"
  by (rule oinvariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_adv]])
  (auto simp: seq1_onl_swap)

lemma rreq_sip:
  "opaadv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onl ΓAODV (λ(σ, l).
      (l ∈ {PAadv-:4, PAadv-:5, PRreq-:0, PRreq-:2} ∧ sip (σ i) ≠ oip (σ i)
        → oip (σ i) ∈ kD(rt (σ (sip (σ i))))
          ∧ nsqn (rt (σ (sip (σ i)))) (oip (σ i)) ≥ osn (σ i)
          ∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ i)) = osn (σ i)
            → (hops (σ i) ≥ the (dhops (rt (σ (sip (σ i)))) (oip (σ i)))
              ∨ the (flag (rt (σ (sip (σ i)))) (oip (σ i))) = inv)))"
    (is "_ ⊨ (?S, ?U →) _")
  proof (inv_terms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh
    adv_wf oaadv_trans]
    onl_oinvariant_sterms [OF adv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
    onl_oinvariant_sterms [OF adv_wf oosn_rreq]
    simp add: seqsimp
    simp del: One_nat_def, rule impI)
  fix σ σ' p l
  assume "(σ, p) ∈ oreachable (opaadv i) ?S ?U"
  and "l ∈ labels ΓAODV p"
  and pre:
    "(l = PAadv-:4 ∨ l = PAadv-:5 ∨ l = PRreq-:0 ∨ l = PRreq-:2) ∧ sip (σ i) ≠ oip (σ i)
      → oip (σ i) ∈ kD (rt (σ (sip (σ i))))
        ∧ osn (σ i) ≤ nsqn (rt (σ (sip (σ i)))) (oip (σ i))
        ∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ i)) = osn (σ i)
          → the (dhops (rt (σ (sip (σ i)))) (oip (σ i))) ≤ hops (σ i)
            ∨ the (flag (rt (σ (sip (σ i)))) (oip (σ i))) = inv)"
  and "other quality_increases {i} σ σ'"
  and hyp: "(l=PAadv-:4 ∨ l=PAadv-:5 ∨ l=PRreq-:0 ∨ l=PRreq-:2) ∧ sip (σ' i) ≠ oip (σ' i)"
  (is "?labels ∧ sip (σ' i) ≠ oip (σ' i)")
  from this(4) have "σ' i = σ i" ..
  with hyp have hyp': "?labels ∧ sip (σ i) ≠ oip (σ i)" by simp
  show "oip (σ' i) ∈ kD (rt (σ' (sip (σ' i))))
    ∧ osn (σ' i) ≤ nsqn (rt (σ' (sip (σ' i)))) (oip (σ' i))
    ∧ (nsqn (rt (σ' (sip (σ' i)))) (oip (σ' i)) = osn (σ' i)
      → the (dhops (rt (σ' (sip (σ' i)))) (oip (σ' i))) ≤ hops (σ' i)
        ∨ the (flag (rt (σ' (sip (σ' i)))) (oip (σ' i))) = inv)"

```

```

proof (cases "sip ( $\sigma$  i) = i")
  assume "sip ( $\sigma$  i)  $\neq$  i"
  from  $\langle$ other quality_increases {i}  $\sigma$   $\sigma'$  $\rangle$ 
    have "quality_increases ( $\sigma$  (sip ( $\sigma$  i))) ( $\sigma'$  (sip ( $\sigma'$  i)))"
      by (rule otherE) (clarsimp simp:  $\langle$ sip ( $\sigma$  i)  $\neq$  i $\rangle$ )
  moreover from  $\langle$ ( $\sigma$ , p)  $\in$  oreachable (opaodv i) ?S ?U $\rangle$  ( $l \in$  labels  $\Gamma_{AODV}$  p) and hyp
    have " $l \leq$  osn ( $\sigma'$  i)"
      by (auto dest!: onl_oinvariant_weakenD [OF oosn_rreq]
          simp add: seqlsimp  $\langle$  $\sigma'$  i =  $\sigma$  i $\rangle$ )
  moreover from  $\langle$ sip ( $\sigma$  i)  $\neq$  i $\rangle$  hyp' and pre
    have "oip ( $\sigma'$  i)  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))"
       $\wedge$  osn ( $\sigma'$  i)  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma'$  i))
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma'$  i))) = osn ( $\sigma'$  i)
       $\longrightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma'$  i)))  $\leq$  hops ( $\sigma'$  i)
       $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma'$  i))) = inv"
    by (auto simp:  $\langle$  $\sigma'$  i =  $\sigma$  i $\rangle$ )
  ultimately show ?thesis
    by (rule quality_increases_rreq_rrep_props)
next
  assume "sip ( $\sigma$  i) = i" thus ?thesis
    using  $\langle$  $\sigma'$  i =  $\sigma$  i $\rangle$  hyp and pre by auto
qed
qed (auto elim!: quality_increases_rreq_rrep_props')

```

lemma odsn\_rrep:

```

"opaodv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onl  $\Gamma_{AODV}$  (seq1 i ( $\lambda$ ( $\xi$ , l).  $l \in$  {PAodv-:6, PAodv-:7}  $\cup$  {PRrep-:n/n. True}  $\longrightarrow$   $l \leq$  dsn  $\xi$ ))"
by (rule oinvariant_weakenE [OF open_seq_invariant [OF dsn_rrep initiali_aodv]])
(auto simp: seq1_onl_swap)

```

lemma rrep\_sip:

```

"opaodv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onl  $\Gamma_{AODV}$  ( $\lambda$ ( $\sigma$ , l).
    ( $l \in$  {PAodv-:6, PAodv-:7, PRrep-:0, PRrep-:1}  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i))
     $\longrightarrow$  dip ( $\sigma$  i)  $\in$  kD(rt ( $\sigma$  (sip ( $\sigma$  i))))
     $\wedge$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))  $\geq$  dsn ( $\sigma$  i)
     $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = dsn ( $\sigma$  i)
     $\longrightarrow$  (hops ( $\sigma$  i)  $\geq$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))))
     $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = inv)))"
(is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")

```

```

proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf
  oadov_trans]
  onl_oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
  onl_oinvariant_sterms [OF aodv_wf odsn_rrep]
  simp del: One_nat_def, rule impI)

```

fix  $\sigma$   $\sigma'$  p l

assume " $(\sigma$ , p)  $\in$  oreachable (opaodv i) ?S ?U"

and " $l \in$  labels  $\Gamma_{AODV}$  p"

and pre:

```

"( $l =$  PAodv-:6  $\vee$   $l =$  PAodv-:7  $\vee$   $l =$  PRrep-:0  $\vee$   $l =$  PRrep-:1)  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i)
 $\longrightarrow$  dip ( $\sigma$  i)  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
 $\wedge$  dsn ( $\sigma$  i)  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))
 $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = dsn ( $\sigma$  i)
 $\longrightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i)))  $\leq$  hops ( $\sigma$  i)
 $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = inv"

```

and "other quality\_increases {i}  $\sigma$   $\sigma'$ "

and hyp: " $(l=$ PAodv-:6  $\vee$   $l=$ PAodv-:7  $\vee$   $l=$ PRrep-:0  $\vee$   $l=$ PRrep-:1)  $\wedge$  sip ( $\sigma'$  i)  $\neq$  dip ( $\sigma'$  i)"  
(is "?labels  $\wedge$  sip ( $\sigma'$  i)  $\neq$  dip ( $\sigma'$  i)")

from this(4) have " $\sigma'$  i =  $\sigma$  i" ..

with hyp have hyp': "?labels  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i)" by simp

show "dip ( $\sigma'$  i)  $\in$  kD (rt ( $\sigma'$  (sip ( $\sigma'$  i))))

$\wedge$  dsn ( $\sigma'$  i)  $\leq$  nsqn (rt ( $\sigma'$  (sip ( $\sigma'$  i)))) (dip ( $\sigma'$  i))

```

    ∧ (nsqn (rt (σ' (sip (σ' i)))) (dip (σ' i)) = dsn (σ' i)
      → the (dhops (rt (σ' (sip (σ' i)))) (dip (σ' i))) ≤ hops (σ' i)
        ∨ the (flag (rt (σ' (sip (σ' i)))) (dip (σ' i))) = inv)"
proof (cases "sip (σ i) = i")
  assume "sip (σ i) ≠ i"
  from ⟨other quality_increases {i} σ σ'⟩
  have "quality_increases (σ (sip (σ i))) (σ' (sip (σ' i)))"
  by (rule otherE) (clarsimp simp: ⟨sip (σ i) ≠ i⟩)
  moreover from ⟨(σ, p) ∈ oreachable (opaadv i) ?S ?U⟩ (l ∈ labels ΓAODV p) and hyp
  have "1 ≤ dsn (σ' i)"
  by (auto dest!: onl_oInvariant_weakenD [OF odsn_rrep]
      simp add: seqlsimp ⟨σ' i = σ i⟩)
  moreover from ⟨sip (σ i) ≠ i⟩ hyp' and pre
  have "dip (σ' i) ∈ kD (rt (σ (sip (σ i))))"
  ∧ dsn (σ' i) ≤ nsqn (rt (σ (sip (σ i)))) (dip (σ' i))
  ∧ (nsqn (rt (σ (sip (σ i)))) (dip (σ' i)) = dsn (σ' i)
    → the (dhops (rt (σ (sip (σ i)))) (dip (σ' i))) ≤ hops (σ' i)
      ∨ the (flag (rt (σ (sip (σ i)))) (dip (σ' i))) = inv)"
  by (auto simp: ⟨σ' i = σ i⟩)
  ultimately show ?thesis
  by (rule quality_increases_rreq_rrep_props)
next
  assume "sip (σ i) = i" thus ?thesis
  using ⟨σ' i = σ i⟩ hyp and pre by auto
qed
qed (auto simp add: seqlsimp elim!: quality_increases_rreq_rrep_props')

```

lemma rerr\_sip:

```

"opaadv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i} →)
  onl ΓAODV (λ(σ, l).
    l ∈ {PAadv-:8, PAadv-:9, PRerr-:0, PRerr-:1}
    → (∀rip∈dom(dests (σ i)). rip∈kD(rt (σ (sip (σ i)))) ∧
      the (dests (σ i) rip) - 1 ≤ nsqn (rt (σ (sip (σ i)))) rip))"
(is "_ ⊨ (?S, ?U →) _")

```

proof -

```

{ fix dests rip sip rsn and σ σ' :: "ip ⇒ state"
  assume qinc: "∀j. quality_increases (σ j) (σ' j)"
  and *: "∀rip∈dom dests. rip ∈ kD (rt (σ sip))
    ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
  and "dests rip = Some rsn"
  from this(3) have "rip∈dom dests" by auto
  with * and ⟨dests rip = Some rsn⟩ have "rip∈kD(rt (σ sip))"
    and "rsn - 1 ≤ nsqn (rt (σ sip)) rip"

```

```

  by (auto dest!: bspec)

```

```

  from qinc have "quality_increases (σ sip) (σ' sip)" ..

```

```

  have "rip ∈ kD(rt (σ' sip)) ∧ rsn - 1 ≤ nsqn (rt (σ' sip)) rip"

```

proof

```

  from ⟨rip∈kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  show "rip ∈ kD(rt (σ' sip))" ..

```

next

```

  from ⟨rip∈kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩

```

```

  have "nsqn (rt (σ sip)) rip ≤ nsqn (rt (σ' sip)) rip" ..

```

```

  with ⟨rsn - 1 ≤ nsqn (rt (σ sip)) rip⟩ show "rsn - 1 ≤ nsqn (rt (σ' sip)) rip"

```

```

  by (rule le_trans)

```

qed

```

} note partial = this

```

show ?thesis

```

  by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
    oaadv_trans]
      onl_oInvariant_sterms [OF aadv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
      other_quality_increases other_localD
      simp del: One_nat_def, intro conjI)

```

(clarsimp simp del: One\_nat\_def split: split\_if\_asm option.split\_asm, erule(2) partial)+  
qed

lemma prerr\_guard: "paadv i  $\models$   
 onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l).$  ( $l = PRerr-:1$   
 $\rightarrow (\forall ip \in \text{dom}(\text{dests } \xi). ip \in vD(\text{rt } \xi)$   
 $\quad \wedge \text{the } (\text{nhop } (\text{rt } \xi) ip) = \text{sip } \xi$   
 $\quad \wedge \text{sqn } (\text{rt } \xi) ip < \text{the } (\text{dests } \xi ip))$ ))"  
 by (inv\_cterms) (clarsimp split: option.split\_asm split\_if\_asm)

lemmas oaddpreRT\_welldefined =  
 open\_seq\_invariant [OF addpreRT\_welldefined initiali\_aadv oaadv\_trans aadv\_trans,  
 simplified seq1\_onl\_swap,  
 THEN oinvariant\_anyact]

lemmas odests\_vD\_inc\_sqn =  
 open\_seq\_invariant [OF dests\_vD\_inc\_sqn initiali\_aadv oaadv\_trans aadv\_trans,  
 simplified seq1\_onl\_swap,  
 THEN oinvariant\_anyact]

lemmas oprerr\_guard =  
 open\_seq\_invariant [OF prerr\_guard initiali\_aadv oaadv\_trans aadv\_trans,  
 simplified seq1\_onl\_swap,  
 THEN oinvariant\_anyact]

Proposition 7.28

lemma seq\_compare\_next\_hop':  
 "opaadv i  $\models$  (otherwith quality\_increases {i} (orecvmsg msg\_fresh),  
 other quality\_increases {i}  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\sigma, \_).$   
 $\quad \forall \text{dip}. \text{let } \text{nhop} = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{dip})$   
 $\quad \text{in } \text{dip} \in \text{kD}(\text{rt } (\sigma i)) \wedge \text{nhop} \neq \text{dip} \rightarrow$   
 $\quad \text{dip} \in \text{kD}(\text{rt } (\sigma \text{nhop})) \wedge \text{nsqn } (\text{rt } (\sigma i)) \text{dip} \leq \text{nsqn } (\text{rt } (\sigma \text{nhop})) \text{dip}$ )"  
 (is "\_  $\models$  (?S, ?U  $\rightarrow$ ) \_")  
 proof -  
 { fix nhop and  $\sigma \sigma' :: "ip \Rightarrow \text{state}"$   
 assume pre: " $\forall \text{dip} \in \text{kD}(\text{rt } (\sigma i)). \text{nhop } \text{dip} \neq \text{dip} \rightarrow$   
 $\quad \text{dip} \in \text{kD}(\text{rt } (\sigma (\text{nhop } \text{dip}))) \wedge \text{nsqn } (\text{rt } (\sigma i)) \text{dip} \leq \text{nsqn } (\text{rt } (\sigma (\text{nhop } \text{dip}))) \text{dip}$ "  
 and qinc: " $\forall j. \text{quality\_increases } (\sigma j) (\sigma' j)$ "  
 have " $\forall \text{dip} \in \text{kD}(\text{rt } (\sigma i)). \text{nhop } \text{dip} \neq \text{dip} \rightarrow$   
 $\quad \text{dip} \in \text{kD}(\text{rt } (\sigma' (\text{nhop } \text{dip}))) \wedge \text{nsqn } (\text{rt } (\sigma i)) \text{dip} \leq \text{nsqn } (\text{rt } (\sigma' (\text{nhop } \text{dip}))) \text{dip}$ "  
 proof (intro ballI impI)  
 fix dip  
 assume "dip  $\in \text{kD}(\text{rt } (\sigma i))$ "  
 and "nhop dip  $\neq$  dip"  
 with pre have "dip  $\in \text{kD}(\text{rt } (\sigma (\text{nhop } \text{dip})))$ "  
 and "nsqn (rt ( $\sigma i$ )) dip  $\leq$  nsqn (rt ( $\sigma (\text{nhop } \text{dip})))$  dip"  
 by auto  
 from qinc have qinc\_nhop: "quality\_increases ( $\sigma (\text{nhop } \text{dip})$ ) ( $\sigma' (\text{nhop } \text{dip})$ )" ..  
 with  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma (\text{nhop } \text{dip}))) \rangle$  have "dip  $\in \text{kD}(\text{rt } (\sigma' (\text{nhop } \text{dip})))$ " ..  
 moreover have "nsqn (rt ( $\sigma i$ )) dip  $\leq$  nsqn (rt ( $\sigma' (\text{nhop } \text{dip})))$  dip"  
 proof -  
 from  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma (\text{nhop } \text{dip}))) \rangle$  qinc\_nhop  
 have "nsqn (rt ( $\sigma (\text{nhop } \text{dip})))$  dip  $\leq$  nsqn (rt ( $\sigma' (\text{nhop } \text{dip})))$  dip" ..  
 with  $\langle \text{nsqn } (\text{rt } (\sigma i)) \text{dip} \leq \text{nsqn } (\text{rt } (\sigma (\text{nhop } \text{dip}))) \text{dip} \rangle$  show ?thesis  
 by simp  
 qed  
 ultimately show "dip  $\in \text{kD}(\text{rt } (\sigma' (\text{nhop } \text{dip})))$   
 $\quad \wedge \text{nsqn } (\text{rt } (\sigma i)) \text{dip} \leq \text{nsqn } (\text{rt } (\sigma' (\text{nhop } \text{dip}))) \text{dip}$ " ..  
 qed  
 } note basic = this

```

{ fix nhop and  $\sigma \sigma' :: "ip \Rightarrow state"$ 
  assume pre: " $\forall dip \in kD(rt (\sigma i)). nhop dip \neq dip \longrightarrow dip \in kD(rt (\sigma (nhop dip)))$ 
     $\wedge nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (nhop dip))) dip"$ 
  and ndest: " $\forall ripc \in dom (dests (\sigma i)). ripc \in kD (rt (\sigma (sip (\sigma i))))$ 
     $\wedge the (dests (\sigma i) ripc) - 1 \leq nsqn (rt (\sigma (sip (\sigma i)))) ripc"$ 
  and issip: " $\forall ip \in dom (dests (\sigma i)). nhop ip = sip (\sigma i)"$ 
  and qinc: " $\forall j. quality\_increases (\sigma j) (\sigma' j)"$ 
  have " $\forall dip \in kD(rt (\sigma i)). nhop dip \neq dip \longrightarrow dip \in kD (rt (\sigma' (nhop dip)))$ 
     $\wedge nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"$ 
  proof (intro ballI impI)
    fix dip
    assume "dip  $\in kD(rt (\sigma i))"$ 
      and "nhop dip  $\neq dip"$ 
    with pre and qinc have "dip  $\in kD(rt (\sigma' (nhop dip)))"$ 
      and "nsqn (rt (\sigma i)) dip  $\leq nsqn (rt (\sigma' (nhop dip))) dip"$ 
    by (auto dest!: basic)

  have "nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip  $\leq nsqn (rt (\sigma' (nhop dip))) dip"$ 
  proof (cases "dip  $\in dom (dests (\sigma i))"$ )
    assume "dip  $\in dom (dests (\sigma i))"$ 
    with  $\langle dip \in kD(rt (\sigma i)) \rangle$  obtain dsn where "dests ( $\sigma i$ ) dip = Some dsn"
      by auto
    with  $\langle dip \in kD(rt (\sigma i)) \rangle$  have "nsqn (invalidate (rt ( $\sigma i$ )) (dests ( $\sigma i$ ))) dip = dsn - 1"
      by (rule nsqn_invalidate_eq)
    moreover have "dsn - 1  $\leq nsqn (rt (\sigma' (nhop dip))) dip"$ 
    proof -
      from  $\langle dests (\sigma i) dip = Some dsn \rangle$  have "the (dests ( $\sigma i$ ) dip) = dsn" by simp
      with ndest and  $\langle dip \in dom (dests (\sigma i)) \rangle$  have "dip  $\in kD (rt (\sigma (sip (\sigma i))))"$ 
        "dsn - 1  $\leq nsqn (rt (\sigma (sip (\sigma i)))) dip"$ 
      by auto
      moreover from issip and  $\langle dip \in dom (dests (\sigma i)) \rangle$  have "nhop dip = sip ( $\sigma i$ )" ..
      ultimately have "dip  $\in kD (rt (\sigma (nhop dip)))"$ 
        and "dsn - 1  $\leq nsqn (rt (\sigma (nhop dip))) dip"$  by auto
      with qinc show "dsn - 1  $\leq nsqn (rt (\sigma' (nhop dip))) dip"$ 
        by simp (metis kD_nsqn_quality_increases_trans)
    qed
    ultimately show ?thesis by simp
  next
  assume "dip  $\notin dom (dests (\sigma i))"$ 
  with  $\langle dip \in kD(rt (\sigma i)) \rangle$ 
    have "nsqn (invalidate (rt ( $\sigma i$ )) (dests ( $\sigma i$ ))) dip = nsqn (rt ( $\sigma i$ )) dip"
      by (rule nsqn_invalidate_other)
  with "nsqn (rt ( $\sigma i$ )) dip  $\leq nsqn (rt (\sigma' (nhop dip))) dip"$  show ?thesis by simp
  qed
  with  $\langle dip \in kD(rt (\sigma' (nhop dip))) \rangle$ 
    show "dip  $\in kD (rt (\sigma' (nhop dip)))"$ 
       $\wedge nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"$  ..
  qed
} note basic_prerr = this

{ fix  $\sigma \sigma' :: "ip \Rightarrow state"$ 
  assume a1: " $\forall dip \in kD(rt (\sigma i)). the (nhop (rt (\sigma i)) dip) \neq dip$ 
     $\longrightarrow dip \in kD(rt (\sigma (the (nhop (rt (\sigma i)) dip))))$ 
     $\wedge nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (the (nhop (rt (\sigma i)) dip)))) dip"$ 
  and a2: " $\forall j. quality\_increases (\sigma j) (\sigma' j)"$ 
  have " $\forall dip \in kD(rt (\sigma i)).$ 
    the (nhop (update (rt ( $\sigma i$ )) (sip ( $\sigma i$ )) (0, unk, val, Suc 0, sip ( $\sigma i$ ), {})) dip)  $\neq dip \longrightarrow$ 
    dip  $\in kD(rt (\sigma' (the (nhop (update (rt (\sigma i)) (sip (\sigma i))$ 
      (0, unk, val, Suc 0, sip ( $\sigma i$ ), {}))
      dip))))  $\wedge$ 
    nsqn (update (rt ( $\sigma i$ )) (sip ( $\sigma i$ )) (0, unk, val, Suc 0, sip ( $\sigma i$ ), {})) dip
     $\leq nsqn (rt (\sigma' (the (nhop (update (rt (\sigma i)) (sip (\sigma i))$ 
      (0, unk, val, Suc 0, sip ( $\sigma i$ ), {}))
      dip))))
```



```

      dip" (is "∀ dip ∈ kD(rt (σ i)). ?P dip")
proof
  fix dip
  assume "dip ∈ kD(rt (σ i))"
  with a1 and a2
    have "the (nhop (rt (σ i)) dip) ≠ dip → dip ∈ kD(rt (σ' (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
    by - (drule(1) basic, auto)
  thus "?P dip" by (cases "dip = sip (σ i)") auto
qed
} note nhop_update_sip = this

{ fix σ σ' oip sip osn hops
  assume pre: "∀ dip ∈ kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip ∈ kD(rt (σ (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
  and qinc: "∀ j. quality_increases (σ j) (σ' j)"
  and *: "sip ≠ oip → oip ∈ kD(rt (σ sip))
    ∧ osn ≤ nsqn (rt (σ sip)) oip
    ∧ (nsqn (rt (σ sip)) oip = osn
      → the (dhops (rt (σ sip)) oip) ≤ hops
      ∨ the (flag (rt (σ sip)) oip) = inv)"
  from pre and qinc
    have pre': "∀ dip ∈ kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
      → dip ∈ kD(rt (σ' (the (nhop (rt (σ i)) dip))))
        ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
    by (rule basic)
  have "(the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }))) oip) ≠ oip
    → oip ∈ kD(rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip, { }))) oip))))
      ∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })) oip
        ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip, { }))) oip)))) oip)"
    (is "?nhop_not_oip → ?oip_in_kD ∧ ?nsqn_le_nsqn")
  proof (rule, split update_rt_split_asm)
    assume "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })"
    and "the (nhop (rt (σ i)) oip) ≠ oip"
    with pre' show "?oip_in_kD ∧ ?nsqn_le_nsqn" by auto
  next
    assume rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })"
    and notoip: ?nhop_not_oip
    with * qinc have ?oip_in_kD
      by auto
    moreover with * pre qinc rtnot notoip have ?nsqn_le_nsqn
      by simp (metis kD_nsqn_quality_increases_trans)
    ultimately show "?oip_in_kD ∧ ?nsqn_le_nsqn" ..
  qed
} note update1 = this

{ fix σ σ' oip sip osn hops
  assume pre: "∀ dip ∈ kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip ∈ kD(rt (σ (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
  and qinc: "∀ j. quality_increases (σ j) (σ' j)"
  and *: "sip ≠ oip → oip ∈ kD(rt (σ sip))
    ∧ osn ≤ nsqn (rt (σ sip)) oip
    ∧ (nsqn (rt (σ sip)) oip = osn
      → the (dhops (rt (σ sip)) oip) ≤ hops
      ∨ the (flag (rt (σ sip)) oip) = inv)"
  from pre and qinc
    have pre': "∀ dip ∈ kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
      → dip ∈ kD(rt (σ' (the (nhop (rt (σ i)) dip))))
        ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
    by (rule basic)

```

```

have "∀ dip ∈ kD(rt (σ i)).
  the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })) dip) ≠ dip
  → dip ∈ kD(rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, { })) dip))))
  ∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })) dip
    ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip, { })) dip)))) dip"
(is "∀ dip ∈ kD(rt (σ i)). _ → ?dip_in_kD dip ∧ ?nsqn_le_nsqn dip")
proof (intro ballI impI, split update_rt_split_asm)
  fix dip
  assume "dip ∈ kD(rt (σ i))"
  and "the (nhop (rt (σ i)) dip) ≠ dip"
  and "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })"
  with pre' show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip" by simp
next
  fix dip
  assume "dip ∈ kD(rt (σ i))"
  and notdip: "the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, { })) dip) ≠ dip"
  and rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })"
  show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip"
  proof (cases "dip = oip")
    assume "dip ≠ oip"
    with pre' (dip ∈ kD(rt (σ i))) notdip
      show ?thesis by clarsimp
  next
    assume "dip = oip"
    with rtnot qinc (dip ∈ kD(rt (σ i))) notdip *
      have "?dip_in_kD dip"
        by simp (metis kD_quality_increases)
    moreover from (dip = oip) rtnot qinc (dip ∈ kD(rt (σ i))) notdip *
      have "?nsqn_le_nsqn dip" by simp (metis kD_nsqn_quality_increases_trans)
    ultimately show ?thesis ..
  qed
qed
} note update2 = this

```

```

have "opaadv i ⊨ (?S, ?U →) onl ΓAODV (λ(σ, _).
  ∀ dip ∈ kD(rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
  → dip ∈ kD(rt (σ (the (nhop (rt (σ i)) dip))))
  ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip)"
by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
  oadv_trans]
  onl_oInvariant_sterms [OF aadv_wf oaddpreRT_welldefined]
  onl_oInvariant_sterms [OF aadv_wf odests_vD_inc_sqn]
  onl_oInvariant_sterms [OF aadv_wf oprerr_guard]
  onl_oInvariant_sterms [OF aadv_wf rreq_sip]
  onl_oInvariant_sterms [OF aadv_wf rrep_sip]
  onl_oInvariant_sterms [OF aadv_wf rerr_sip]
  other_quality_increases
  other_localD
  solve: basic basic_prerr
  simp add: seqsimp_nsqn_invalidate_nhop_update_sip
  simp del: One_nat_def)
(rule conjI, erule(2) update1, erule(2) update2)+

```

```

thus ?thesis unfolding Let_def by auto
qed

```

Proposition 7.30

```

lemmas okD_unk_or_atleast_one =
  open_seq_invariant [OF kD_unk_or_atleast_one initiali_aadv,
    simplified seq1_onl_swap]

```

```

lemmas ozero_seq_unk_hops_one =
  open_seq_invariant [OF zero_seq_unk_hops_one initiali_aadv,
    simplified seq1_onl_swap]

```

```

lemma oreachable_fresh_okD_unk_or_atleast_one:

```

```

  fixes dip
  assumes "(σ, p) ∈ oreachable (opaadv i)
    (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
      ∧ msg_zhops m)))
    (other quality_increases {i})"
  and "dip ∈ kD(rt (σ i))"
  shows "π3(the (rt (σ i) dip)) = unk ∨ 1 ≤ π2(the (rt (σ i) dip))"
  (is "?P dip")
  proof -
  have "∃l. l ∈ labels ΓAODV p" by (metis aadv_ex_label)
  with assms(1) have "∀dip ∈ kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaadv_trans aadv_trans]],
    auto dest!: otherwith_actionD onlD simp: seq1simp)
  with ⟨dip ∈ kD(rt (σ i))⟩ show ?thesis by simp
  qed

```

```

lemma oreachable_fresh_ozero_seq_unk_hops_one:

```

```

  fixes dip
  assumes "(σ, p) ∈ oreachable (opaadv i)
    (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
      ∧ msg_zhops m)))
    (other quality_increases {i})"
  and "dip ∈ kD(rt (σ i))"
  shows "sqn (rt (σ i) dip) = 0 →
    sqnf (rt (σ i) dip) = unk
    ∧ the (dhops (rt (σ i) dip)) = 1
    ∧ the (nhop (rt (σ i) dip)) = dip"
  (is "?P dip")
  proof -
  have "∃l. l ∈ labels ΓAODV p" by (metis aadv_ex_label)
  with assms(1) have "∀dip ∈ kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaadv_trans aadv_trans]],
    auto dest!: onlD otherwith_actionD simp: seq1simp)
  with ⟨dip ∈ kD(rt (σ i))⟩ show ?thesis by simp
  qed

```

```

lemma seq_nhop_quality_increases':

```

```

  shows "opaadv i ⊨ (otherwith (op=) {i}
    (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} →)
    onl ΓAODV (λ(σ, _). ∀dip. let nhip = the (nhop (rt (σ i) dip))
      in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip))
      ∧ nhip ≠ dip
      → (rt (σ i) ⊑dip rt (σ nhip)))"
  (is "_ ⊨ (?S i, _ →) _")

```

```

  proof -

```

```

    have weaken:

```

```

      "∧P I Q R P. p ⊨ (otherwith quality_increases I (orecvmsg Q), other quality_increases I →) P
      ⇒ p ⊨ (otherwith (op=) I (orecvmsg (λσ m. Q σ m ∧ R σ m)), other quality_increases I →) P"
    by auto

```

```

  {

```

```

    fix i a and σ σ' :: "ip ⇒ state"

```

```

    assume a1: "∀dip. dip ∈ vD(rt (σ i))

```

```

      ∧ dip ∈ vD(rt (σ (the (nhop (rt (σ i) dip))))))

```

```

      ∧ (the (nhop (rt (σ i) dip)) ≠ dip

```

```

        → rt (σ i) ⊑dip rt (σ (the (nhop (rt (σ i) dip))))"

```

```

    and ow: "?S i σ σ' a"

```

```

    have "∀dip. dip ∈ vD(rt (σ i))

```

```

      ∧ dip ∈ vD (rt (σ' (the (nhop (rt (σ i) dip))))))

```

```

      ∧ (the (nhop (rt (σ i)) dip)) ≠ dip
    → rt (σ i) ⊆dip rt (σ' (the (nhop (rt (σ i)) dip))))"
proof clarify
  fix dip
  assume a2: "dip ∈ vD(rt (σ i))"
    and a3: "dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))"
    and a4: "(the (nhop (rt (σ i)) dip)) ≠ dip"
  from ow have "∀ j. j ≠ i → σ j = σ' j" by auto
  show "rt (σ i) ⊆dip rt (σ' (the (nhop (rt (σ i)) dip))))"
  proof (cases "(the (nhop (rt (σ i)) dip)) = i")
    assume "(the (nhop (rt (σ i)) dip)) = i"
    with ⟨dip ∈ vD(rt (σ i))⟩ have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))" by simp
    with a1 a2 a4 have "rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp
    with ⟨(the (nhop (rt (σ i)) dip)) = i⟩ have "rt (σ i) ⊆dip rt (σ i)" by simp
    hence False by simp
    thus ?thesis ..
  next
  assume "(the (nhop (rt (σ i)) dip)) ≠ i"
  with ⟨∀ j. j ≠ i → σ j = σ' j⟩
    have *: "σ (the (nhop (rt (σ i)) dip)) = σ' (the (nhop (rt (σ i)) dip))" by simp
  with ⟨dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))⟩
    have "dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))" by simp
  with a1 a2 a4 have "rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp
  with * show ?thesis by simp
qed
qed
} note basic = this

{ fix σ σ' a dip sip i
  assume a1: "∀ dip. dip ∈ vD(rt (σ i))
    ∧ dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))"
  and ow: "?S i σ σ' a"
  have "∀ dip. dip ∈ vD(update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {}))
    ∧ dip ∈ vD(rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip))))
    ∧ the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip) ≠ dip
    → update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})
      ⊆dip rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip)))"
  proof clarify
    fix dip
    assume a2: "dip ∈ vD (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {}))"
      and a3: "dip ∈ vD(rt (σ' (the (nhop
        (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip))))"
      and a4: "the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip) ≠ dip"
    show "update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})
      ⊆dip rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip)))"
    proof (cases "dip = sip")
      assume "dip = sip"
      with ⟨the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip) ≠ dip⟩
        have False by simp
      thus ?thesis ..
    next
      assume [simp]: "dip ≠ sip"
      from a2 have "dip ∈ vD(rt (σ i)) ∨ dip = sip"
        by (rule vD_update_val)
      with ⟨dip ≠ sip⟩ have "dip ∈ vD(rt (σ i))" by simp
      moreover from a3 have "dip ∈ vD(rt (σ' (the (nhop (rt (σ i)) dip))))" by simp
      moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
      ultimately have "rt (σ i) ⊆dip rt (σ' (the (nhop (rt (σ i)) dip)))"
        using a1 ow by - (drule(1) basic, simp)
      with ⟨dip ≠ sip⟩ show ?thesis
        by - (erule rt_strictly_fresher_update_other, simp)
    qed
  qed
}

```

```

qed
} note update_0_unk = this

{ fix  $\sigma$  a  $\sigma'$  nhop
  assume pre: " $\forall \text{dip. dip} \in \text{vD}(\text{rt}(\sigma i)) \wedge \text{dip} \in \text{vD}(\text{rt}(\sigma(\text{nhop dip}))) \wedge \text{nhop dip} \neq \text{dip}$ 
     $\rightarrow \text{rt}(\sigma i) \sqsubseteq_{\text{dip}} \text{rt}(\sigma(\text{nhop dip}))$ "
    and ow: "?S i  $\sigma$   $\sigma'$  a"
  have " $\forall \text{dip. dip} \in \text{vD}(\text{invalidate}(\text{rt}(\sigma i))(\text{dests}(\sigma i)))$ 
     $\wedge \text{dip} \in \text{vD}(\text{rt}(\sigma'(\text{nhop dip}))) \wedge \text{nhop dip} \neq \text{dip}$ 
     $\rightarrow \text{rt}(\sigma i) \sqsubseteq_{\text{dip}} \text{rt}(\sigma'(\text{nhop dip}))$ "
proof clarify
  fix dip
  assume "dip  $\in$  vD(invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i)))"
    and "dip  $\in$  vD(rt ( $\sigma'$  (nhop dip)))"
    and "nhop dip  $\neq$  dip"
  from this(1) have "dip  $\in$  vD (rt ( $\sigma$  i))"
    by (clarsimp dest!: vD_invalidate_vD_not_dests)
  moreover from ow have " $\forall j. j \neq i \rightarrow \sigma j = \sigma' j$ " by auto
  ultimately have "rt ( $\sigma$  i)  $\sqsubseteq_{\text{dip}}$  rt ( $\sigma$  (nhop dip))"
    using pre (dip  $\in$  vD (rt ( $\sigma'$  (nhop dip)))) (nhop dip  $\neq$  dip)
    by metis
  with (forall j. j  $\neq$  i  $\rightarrow$   $\sigma$  j =  $\sigma'$  j) show "rt ( $\sigma$  i)  $\sqsubseteq_{\text{dip}}$  rt ( $\sigma'$  (nhop dip))"
    by (metis rt_strictly_fresher_irefl)
qed
} note invalidate = this

{ fix  $\sigma$  a  $\sigma'$  dip oip osn sip hops i
  assume pre: " $\forall \text{dip. dip} \in \text{vD}(\text{rt}(\sigma i))$ 
     $\wedge \text{dip} \in \text{vD}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$ 
     $\wedge \text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}$ 
     $\rightarrow \text{rt}(\sigma i) \sqsubseteq_{\text{dip}} \text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))$ "
    and ow: "?S i  $\sigma$   $\sigma'$  a"
    and "Suc 0  $\leq$  osn"
    and a6: "sip  $\neq$  oip  $\rightarrow$  oip  $\in$  kD (rt ( $\sigma$  sip))
       $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma$  sip)) oip
       $\wedge$  (nsqn (rt ( $\sigma$  sip)) oip = osn
         $\rightarrow$  the (dhops (rt ( $\sigma$  sip)) oip)  $\leq$  hops
           $\vee$  the (flag (rt ( $\sigma$  sip)) oip) = inv)"
  and after: " $\sigma' i = \sigma i$  (rt := update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { })))"
  have " $\forall \text{dip. dip} \in \text{vD}(\text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip}, \{ \}))$ 
     $\wedge \text{dip} \in \text{vD}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip}, \{ \})) \text{dip}))))$ 
     $\wedge \text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip}, \{ \})) \text{dip}) \neq \text{dip}$ 
     $\rightarrow \text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip}, \{ \})$ 
       $\sqsubseteq_{\text{dip}}$ 
      rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { }))) dip)))"
proof clarify
  fix dip
  assume a2: "dip  $\in$  vD(update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc (hops), sip, { })))"
    and a3: "dip  $\in$  vD(rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { }))) dip))))"
    and a4: "the (nhop (update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { }))) dip)  $\neq$  dip"
  from ow have a5: " $\forall j. j \neq i \rightarrow \sigma j = \sigma' j$ " by auto
  show "update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { })
     $\sqsubseteq_{\text{dip}}$ 
    rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { }))) dip)))"
    (is "?rt1  $\sqsubseteq_{\text{dip}}$  ?rt2 dip")
proof (cases "?rt1 = rt ( $\sigma$  i)")
  assume nochange [simp]:
    "update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { }) = rt ( $\sigma$  i)"

  from after have " $\sigma' i = \sigma i$ " by simp
  with a5 have " $\forall j. \sigma j = \sigma' j$ " by metis

```

```

from a2 have "dip∈vD (rt (σ i))" by simp
moreover from a3 have "dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))"
  using nochange and ⟨∀j. σ j = σ' j⟩ by clarsimp
moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) ⊑dip rt (σ (the (nhop (rt (σ i)) dip)))"
  using pre by simp

hence "rt (σ i) ⊑dip rt (σ' (the (nhop (rt (σ i)) dip)))"
  using ⟨∀j. σ j = σ' j⟩ by simp
thus "?thesis" by simp
next
assume change: "?rt1 ≠ rt (σ i)"
from after a2 have "dip∈kD(rt (σ' i))" by auto
show ?thesis
proof (cases "dip = oip")
  assume "dip ≠ oip"

  with a2 have "dip∈vD (rt (σ i))" by auto
  moreover with a3 a5 after and ⟨dip ≠ oip⟩
    have "dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))"
      by simp metis
  moreover from a4 and ⟨dip ≠ oip⟩ have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
  ultimately have "rt (σ i) ⊑dip rt (σ (the (nhop (rt (σ i)) dip)))"
    using pre by simp

  with after and a5 and ⟨dip ≠ oip⟩ show ?thesis
    by simp (metis rt_strictly_fresher_update_other
      rt_strictly_fresher_irefl)
next
assume "dip = oip"

with a4 and change have "sip ≠ oip" by simp
with a6 have "oip∈kD(rt (σ sip))"
  and "osn ≤ nsqn (rt (σ sip)) oip" by auto

from a3 change ⟨dip = oip⟩ have "oip∈vD(rt (σ' sip))" by simp
hence "the (flag (rt (σ' sip)) oip) = val" by simp

from ⟨oip∈kD(rt (σ sip))⟩
have "osn < nsqn (rt (σ' sip)) oip ∨ (osn = nsqn (rt (σ' sip)) oip
  ∧ the (dhops (rt (σ' sip)) oip) ≤ hops)"
proof
  assume "oip∈vD(rt (σ sip))"
  hence "the (flag (rt (σ sip)) oip) = val" by simp
  with a6 ⟨sip ≠ oip⟩ have "nsqn (rt (σ sip)) oip = osn →
    the (dhops (rt (σ sip)) oip) ≤ hops"
    by simp
  show ?thesis
proof (cases "sip = i")
  assume "sip ≠ i"
  with a5 have "σ sip = σ' sip" by simp
  with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
    and ⟨nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
  show ?thesis by auto
next
— alternative to using sip_not_ip
assume [simp]: "sip = i"
have "?rt1 = rt (σ i)"
proof (rule update_cases_kD, simp_all)
  from ⟨Suc 0 ≤ osn⟩ show "0 < osn" by simp
next
  from ⟨oip∈kD(rt (σ sip))⟩ and ⟨sip = i⟩ show "oip∈kD(rt (σ i))"
    by simp
next

```

```

assume "sqn (rt (σ i)) oip < osn"
also from ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
  have "... ≤ nsqn (rt (σ i)) oip" by simp
also have "... ≤ sqn (rt (σ i)) oip"
  by (rule nsqn_sqn)
finally have "sqn (rt (σ i)) oip < sqn (rt (σ i)) oip" .
hence False by simp
thus "(λa. if a = oip
  then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
  else rt (σ i) a) = rt (σ i) a" ..
next
assume "sqn (rt (σ i)) oip = osn"
  and "Suc hops < the (dhops (rt (σ i)) oip)"
from this(1) and ⟨oip ∈ vD (rt (σ sip))⟩ have "nsqn (rt (σ i)) oip = osn"
  by simp
with ⟨nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
  have "the (dhops (rt (σ i)) oip) ≤ hops" by simp
with ⟨Suc hops < the (dhops (rt (σ i)) oip)⟩ have False by simp
thus "(λa. if a = oip
  then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
  else rt (σ i) a) = rt (σ i) a" ..
next
assume "the (flag (rt (σ i)) oip) = inv"
with ⟨the (flag (rt (σ sip)) oip) = val⟩ have False by simp
thus "(λa. if a = oip
  then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
  else rt (σ i) a) = rt (σ i) a" ..
next
from ⟨oip ∈ kD (rt (σ sip))⟩
  show "(λa. if a = oip then Some (the (rt (σ i) oip)) else rt (σ i) a) = rt (σ i) a"
  by (auto dest!: kD_Some)
qed
with change have False ..
thus ?thesis ..
qed
next
assume "oip ∈ iD (rt (σ sip))"
with ⟨the (flag (rt (σ' sip)) oip) = val⟩ and a5 have "sip = i"
  by (metis f.distinct(1) iD_flag_is_inv)
from ⟨oip ∈ iD (rt (σ sip))⟩ have "the (flag (rt (σ sip)) oip) = inv" by auto
with ⟨sip = i⟩ ⟨Suc 0 ≤ osn⟩ change after ⟨oip ∈ kD (rt (σ sip))⟩
  have "nsqn (rt (σ sip)) oip < nsqn (rt (σ' sip)) oip"
  unfolding update_def
  by (clarsimp split: option.split_asm split_if_asm)
  (auto simp: sqn_def)
with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩ have "osn < nsqn (rt (σ' sip)) oip"
  by simp
thus ?thesis ..
qed
thus ?thesis
proof
assume osnlt: "osn < nsqn (rt (σ' sip)) oip"
from ⟨dip ∈ kD (rt (σ' i))⟩ and ⟨dip = oip⟩ have "dip ∈ kD (?rt1)" by simp
moreover from a3 have "dip ∈ kD (?rt2 dip)" by simp
moreover have "nsqn ?rt1 dip < nsqn (?rt2 dip) dip"
  proof -
  have "nsqn ?rt1 oip = osn"
    by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
  also have "... < nsqn (rt (σ' sip)) oip" using osnlt .
  also have "... = nsqn (?rt2 oip) oip" by (simp add: change)
  finally show ?thesis
    using ⟨dip = oip⟩ by simp
  qed
ultimately show ?thesis

```

```

    by (rule rt_strictly_fresher_ltI)
next
assume osneg: "osn = nsqn (rt (σ' sip)) oip ∧ the (dhops (rt (σ' sip)) oip) ≤ hops"

have "oip ∈ kD(?rt1)" by simp
moreover from a3 ⟨dip = oip⟩ have "oip ∈ kD(?rt2 oip)" by simp

moreover have "nsqn ?rt1 oip = nsqn (?rt2 oip) oip"
proof -
  from osneg have "osn = nsqn (rt (σ' sip)) oip" ..
  also have "osn = nsqn ?rt1 oip"
    by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
  also have "nsqn (rt (σ' sip)) oip = nsqn (?rt2 oip) oip"
    by (simp add: change)
  finally show ?thesis .
qed

moreover have "π₅(the (?rt2 oip oip)) < π₅(the (?rt1 oip))"
proof -
  from osneg have "the (dhops (rt (σ' sip)) oip) ≤ hops" ..
  moreover from ⟨oip ∈ vD (rt (σ' sip))⟩ have "oip ∈ kD(rt (σ' sip))" by auto
  ultimately have "π₅(the (rt (σ' sip) oip)) ≤ hops"
    by (auto simp add: proj5_eq_dhops)
  also from change after have "hops < π₅(the (rt (σ' i) oip))"
    by (simp add: proj5_eq_dhops) (metis dhops_update_changed_lessI)
  finally have "π₅(the (rt (σ' sip) oip)) < π₅(the (rt (σ' i) oip))" .
  with change after show ?thesis by simp
qed

ultimately have "?rt1 □oip ?rt2 oip"
  by (rule rt_strictly_fresher_eqI)
with ⟨dip = oip⟩ show ?thesis by simp
qed
qed
qed
} note rreq_rrep_update = this

have "opaadv i ⊨ (otherwith (op=) {i}) (orecvmsg (λσ m. msg_fresh σ m
                                                ∧ msg_zhops m)),
      other quality_increases {i} →)
  onl ΓAODV
  (λ(σ, _). ∀dip. dip ∈ vD (rt (σ i)) ∩ vD (rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) □dip rt (σ (the (nhop (rt (σ i)) dip))))"
proof (inv_cterms inv add: onl_oinvariant_sterms [OF aadv_wf rreq_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf rrep_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf rerr_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf oosn_rreq [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf odsn_rrep [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf oaddpreRT_welldefined])
  solve: basic update_0_unk invalidate rreq_rrep_update
  simp add: seqlsimp)
fix σ σ' p l
assume or: "(σ, p) ∈ oreachable (opaadv i) (?S i) (other quality_increases {i})"
  and "other quality_increases {i} σ σ'"
  and ll: "l ∈ labels ΓAODV p"
  and pre: "∀dip. dip ∈ vD (rt (σ i))
            ∧ dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))
            ∧ the (nhop (rt (σ i)) dip) ≠ dip
            → rt (σ i) □dip rt (σ (the (nhop (rt (σ i)) dip))))"
from this(1-2)
  have or': "(σ', p) ∈ oreachable (opaadv i) (?S i) (other quality_increases {i})"
    by - (rule oreachable_other')

```



from or and ll have next\_hop: " $\forall$ dip. let nhip = the (nhop (rt ( $\sigma$  i)) dip)  
in dip  $\in$  kD(rt ( $\sigma$  i))  $\wedge$  nhip  $\neq$  dip  
 $\rightarrow$  dip  $\in$  kD(rt ( $\sigma$  nhip))  
 $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma$  nhip)) dip"  
by (auto dest!: onl\_oInvariant\_weakenD [OF seq\_compare\_next\_hop'])

from or and ll have unk\_hops\_one: " $\forall$ dip $\in$ kD (rt ( $\sigma$  i)). sqn (rt ( $\sigma$  i)) dip = 0  
 $\rightarrow$  sqnf (rt ( $\sigma$  i)) dip = unk  
 $\wedge$  the (dhops (rt ( $\sigma$  i)) dip) = 1  
 $\wedge$  the (nhop (rt ( $\sigma$  i)) dip) = dip"  
by (auto dest!: onl\_oInvariant\_weakenD [OF ozero\_seq\_unk\_hops\_one  
[OF oaadv\_trans aadv\_trans]]  
otherwith\_actionD  
simp: seqsimp)

from <other quality\_increases {i}  $\sigma$   $\sigma'$ > have " $\sigma' i = \sigma i$ " by auto  
hence "quality\_increases ( $\sigma$  i) ( $\sigma' i$ )" by auto  
with <other quality\_increases {i}  $\sigma$   $\sigma'$ > have " $\forall j$ . quality\_increases ( $\sigma j$ ) ( $\sigma' j$ )"  
by - (erule otherE, metis singleton\_iff)

show " $\forall$ dip. dip  $\in$  vD (rt ( $\sigma' i$ ))  
 $\wedge$  dip  $\in$  vD (rt ( $\sigma'$  (the (nhop (rt ( $\sigma' i$ )) dip))))  
 $\wedge$  the (nhop (rt ( $\sigma' i$ )) dip)  $\neq$  dip  
 $\rightarrow$  rt ( $\sigma' i$ )  $\sqsubset_{\text{dip}}$  rt ( $\sigma'$  (the (nhop (rt ( $\sigma' i$ )) dip)))"

proof clarify

fix dip  
assume "dip $\in$ vD(rt ( $\sigma' i$ ))"  
and "dip $\in$ vD(rt ( $\sigma'$  (the (nhop (rt ( $\sigma' i$ )) dip))))"  
and "the (nhop (rt ( $\sigma' i$ )) dip)  $\neq$  dip"  
from this(1) and < $\sigma' i = \sigma i$ > have "dip $\in$ vD(rt ( $\sigma i$ ))"  
and "dip $\in$ kD(rt ( $\sigma i$ ))"  
by auto

from <the (nhop (rt ( $\sigma' i$ )) dip)  $\neq$  dip> and < $\sigma' i = \sigma i$ >  
have "the (nhop (rt ( $\sigma i$ )) dip)  $\neq$  dip" (is "?nhip  $\neq$  \_") by simp  
with <dip $\in$ kD(rt ( $\sigma i$ ))> and next\_hop  
have "dip $\in$ kD(rt ( $\sigma$  (?nhip)))"  
and nsqns: "nsqn (rt ( $\sigma i$ )) dip  $\leq$  nsqn (rt ( $\sigma$  ?nhip)) dip"  
by (auto simp: Let\_def)

have "0 < sqn (rt ( $\sigma i$ )) dip"  
proof (rule neq0\_conv [THEN iffD1, OF notI])  
assume "sqn (rt ( $\sigma i$ )) dip = 0"  
with <dip $\in$ kD(rt ( $\sigma i$ ))> and unk\_hops\_one  
have "?nhip = dip" by simp  
with <?nhip  $\neq$  dip> show False ..  
qed

also have "... = nsqn (rt ( $\sigma i$ )) dip"  
by (rule vD\_nsqn\_sqn [OF <dip $\in$ vD(rt ( $\sigma i$ ))>, THEN sym])  
also have "...  $\leq$  nsqn (rt ( $\sigma$  ?nhip)) dip"  
by (rule nsqns)  
also have "...  $\leq$  sqn (rt ( $\sigma$  ?nhip)) dip"  
by (rule nsqn\_sqn)  
finally have "0 < sqn (rt ( $\sigma$  ?nhip)) dip" .

have "rt ( $\sigma i$ )  $\sqsubset_{\text{dip}}$  rt ( $\sigma'$  ?nhip)"  
proof (cases "dip $\in$ vD(rt ( $\sigma$  ?nhip))")  
assume "dip $\in$ vD(rt ( $\sigma$  ?nhip))"  
with pre <dip $\in$ vD(rt ( $\sigma i$ ))> and <?nhip  $\neq$  dip>  
have "rt ( $\sigma i$ )  $\sqsubset_{\text{dip}}$  rt ( $\sigma$  ?nhip)" by auto  
moreover from < $\forall j$ . quality\_increases ( $\sigma j$ ) ( $\sigma' j$ )>  
have "quality\_increases ( $\sigma$  ?nhip) ( $\sigma'$  ?nhip)" ..  
ultimately show ?thesis

```

    using ⟨dip∈kD(rt (σ ?nhip))⟩
    by (rule strictly_fresher_quality_increases_right)
next
assume "dip∉vD(rt (σ ?nhip))"
with ⟨dip∈kD(rt (σ ?nhip))⟩ have "dip∈iD(rt (σ ?nhip))" ..
hence "the (flag (rt (σ ?nhip)) dip) = inv"
  by auto
have "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ ?nhip)) dip"
  by (rule nsqns)
also from ⟨dip∈iD(rt (σ ?nhip))⟩
  have "... = sqn (rt (σ ?nhip)) dip - 1" ..
also have "... < sqn (rt (σ' ?nhip)) dip"
  proof -
    from ⟨∀j. quality_increases (σ j) (σ' j)⟩
      have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
    hence "∀ip. sqn (rt (σ ?nhip)) ip ≤ sqn (rt (σ' ?nhip)) ip" by auto
    hence "sqn (rt (σ ?nhip)) dip ≤ sqn (rt (σ' ?nhip)) dip" ..
    with ⟨0 < sqn (rt (σ ?nhip)) dip⟩ show ?thesis by auto
  qed
also have "... = nsqn (rt (σ' ?nhip)) dip"
  proof (rule vD_nsqn_sqn [THEN sym])
    from ⟨dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and ⟨σ' i = σ i⟩
      show "dip∈vD(rt (σ' ?nhip))" by simp
  qed
  finally have "nsqn (rt (σ i)) dip < nsqn (rt (σ' ?nhip)) dip" .

moreover from ⟨dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and ⟨σ' i = σ i⟩
  have "dip∈kD(rt (σ' ?nhip))" by auto
ultimately show "rt (σ i) ⊑dip rt (σ' ?nhip)"
  using ⟨dip∈kD(rt (σ i))⟩ by - (rule rt_strictly_fresher_ltI)
qed
with ⟨σ' i = σ i⟩ show "rt (σ' i) ⊑dip rt (σ' (the (nhop (rt (σ' i)) dip)))"
  by simp
qed
qed
thus ?thesis unfolding Let_def .
qed

```

lemma seq\_nhop\_quality\_increases:

```

shows "opaadv i ⊨ (otherwith (op=) {i}
  (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
  other quality_increases {i} →)
  global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊑dip (rt (σ nhip)))"
  by (rule oinvariant_weakenE [OF seq_nhop_quality_increases']) (auto dest!: onID)

```

end

## 0.11 Routing graphs and loop freedom

```

theory Loop_Freedom
imports Aadv_Predicates Fresher
begin

```

Define the central theorem that relates an invariant over network states to the absence of loops in the associate routing graph.

**definition**

```

rt_graph :: "(ip ⇒ state) ⇒ ip ⇒ ip rel"
where
  "rt_graph σ = (λdip.
  {(ip, ip') | ip ip' dsn dsk hops pre.
  ip ≠ dip ∧ rt (σ ip) dip = Some (dsn, dsk, val, hops, ip', pre)})"

```

Given the state of a network  $\sigma$ , a routing graph for a given destination ip address  $dip$  abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

```
lemma rt_graphE [elim]:
  fixes n dip ip ip'
  assumes "(ip, ip') ∈ rt_graph σ dip"
  shows "ip ≠ dip ∧ (∃r. rt (σ ip) = r
    ∧ (∃dsn dsk hops pre. r dip = Some (dsn, dsk, val, hops, pre)))"
  using assms unfolding rt_graph_def by auto
```

```
lemma rt_graph_vD [dest]:
  "∧ip ip' σ dip. (ip, ip') ∈ rt_graph σ dip ⇒ dip ∈ vD(rt (σ ip))"
  unfolding rt_graph_def vD_def by auto
```

```
lemma rt_graph_vD_trans [dest]:
  "∧ip ip' σ dip. (ip, ip') ∈ (rt_graph σ dip)+ ⇒ dip ∈ vD(rt (σ ip))"
  by (erule converse_tranclE) auto
```

```
lemma rt_graph_not_dip [dest]:
  "∧ip ip' σ dip. (ip, ip') ∈ rt_graph σ dip ⇒ ip ≠ dip"
  unfolding rt_graph_def by auto
```

```
lemma rt_graph_not_dip_trans [dest]:
  "∧ip ip' σ dip. (ip, ip') ∈ (rt_graph σ dip)+ ⇒ ip ≠ dip"
  by (erule converse_tranclE) auto
```

NB: the property below cannot be lifted to the transitive closure

```
lemma rt_graph_nhip_is_nhop [dest]:
  "∧ip ip' σ dip. (ip, ip') ∈ rt_graph σ dip ⇒ ip' = the (nhop (rt (σ ip)) dip)"
  unfolding rt_graph_def by auto
```

```
theorem inv_to_loop_freedom:
  assumes "∀i dip. let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊆dip (rt (σ nhip))"
  shows "∀dip. irrefl ((rt_graph σ dip)+)"
  using assms proof (intro allI)
  fix σ :: "ip ⇒ state" and dip
  assume inv: "∀ip dip.
    let nhip = the (nhop (rt (σ ip)) dip)
    in dip ∈ vD(rt (σ ip)) ∩ vD(rt (σ nhip)) ∧
      nhip ≠ dip → rt (σ ip) ⊆dip rt (σ nhip)"
  { fix ip ip'
    assume "(ip, ip') ∈ (rt_graph σ dip)+"
      and "dip ∈ vD(rt (σ ip'))"
      and "ip' ≠ dip"
    hence "rt (σ ip) ⊆dip rt (σ ip'"
    proof induction
      fix nhip
      assume "(ip, nhip) ∈ rt_graph σ dip"
        and "dip ∈ vD(rt (σ nhip))"
        and "nhip ≠ dip"
      from ⟨(ip, nhip) ∈ rt_graph σ dip⟩ have "dip ∈ vD(rt (σ ip))"
        and "nhip = the (nhop (rt (σ ip)) dip)"
        by auto
      from ⟨dip ∈ vD(rt (σ ip))⟩ and ⟨dip ∈ vD(rt (σ nhip))⟩
        have "dip ∈ vD(rt (σ ip)) ∩ vD(rt (σ nhip))" ..
      with ⟨nhip = the (nhop (rt (σ ip)) dip)⟩
        and ⟨nhip ≠ dip⟩
        and inv
        show "rt (σ ip) ⊆dip rt (σ nhip)"
        by (clarsimp simp: Let_def)
    next
      fix nhip nhip'
      assume "(ip, nhip) ∈ (rt_graph σ dip)+"
```

```

    and "(nhip, nhip') ∈ rt_graph σ dip"
    and IH: "[[ dip ∈ vD(rt (σ nhip)); nhip ≠ dip ]] ⇒ rt (σ ip) ⊑dip rt (σ nhip)"
    and "dip ∈ vD(rt (σ nhip'))"
    and "nhip' ≠ dip"
  from ⟨(nhip, nhip') ∈ rt_graph σ dip⟩ have 1: "dip ∈ vD(rt (σ nhip))"
    and 2: "nhip' ≠ dip"
    and "nhip' = the (nhop (rt (σ nhip)) dip)"

  by auto
  from 1 2 have "rt (σ ip) ⊑dip rt (σ nhip)" by (rule IH)
  also have "rt (σ nhip) ⊑dip rt (σ nhip'"
  proof -
    from ⟨dip ∈ vD(rt (σ nhip))⟩ and ⟨dip ∈ vD(rt (σ nhip'))⟩
    have "dip ∈ vD(rt (σ nhip)) ∩ vD(rt (σ nhip'))" ..
    with ⟨nhip' ≠ dip⟩
    and ⟨nhip' = the (nhop (rt (σ nhip)) dip)⟩
    and inv
    show "rt (σ nhip) ⊑dip rt (σ nhip'"
    by (clarsimp simp: Let_def)
  qed
  finally show "rt (σ ip) ⊑dip rt (σ nhip'" .
qed } note fresher = this

```

```

show "irrefl ((rt_graph σ dip)+)"
unfolding irrefl_def proof (intro allI notI)
  fix ip
  assume "(ip, ip) ∈ (rt_graph σ dip)+"
  moreover then have "dip ∈ vD(rt (σ ip))"
    and "ip ≠ dip"

  by auto
  ultimately have "rt (σ ip) ⊑dip rt (σ ip)" by (rule fresher)
  thus False by simp
qed
qed

```

end

## 0.12 Lift and transfer invariants to show loop freedom

```

theory Aadv_Loop_Freedom
imports "../AWN/OClosed_Transfer" "../AWN/Qmsg_Lifting" Global_Invariants Loop_Freedom
begin

```

### 0.12.1 Lift to parallel processes with queues

```

lemma par_step_no_change_on_send_or_receive:
  fixes σ s a σ' s'
  assumes "((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos ΓAODV i) (seqp_sos ΓQMSG)"
  and "a ≠ τ"
  shows "σ' i = σ i"
  using assms by (rule qmsg_no_change_on_send_or_receive)

lemma par_nhop_quality_increases:
  shows "opaadv i ⟨⟨i qmsg ⊢ (otherwith (op=) {i} (orecvmsg (λσ m.
    msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} →)
  global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊑dip (rt (σ nhip)))⟩⟩"

proof (rule lift_into_qmsg [OF seq_nhop_quality_increases])
show "opaadv i ⊢A (otherwith (op=) {i}
  (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
  other quality_increases {i} →)
  globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
proof (rule ostep_invariant_weakenE [OF quality_increases], simp_all)

```

```

fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
assume "onll ΓAODV (λ((σ, _), _, (σ', _)). ∀j. quality_increases (σ j) (σ' j)) t"
thus "quality_increases (fst (fst t) i) (fst (snd (snd t)) i)"
  by (cases t) (clarsimp dest!: onllD, metis aadv_ex_label)
next
fix σ σ' a
assume "otherwith (op=) {i}
      (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)) σ σ' a"
thus "otherwith quality_increases {i} (orecvmsg (λ_. rreq_rrep_sn)) σ σ' a"
  by - (erule weaken_otherwith, auto)
qed
qed auto

lemma par_rreq_rrep_sn_quality_increases:
"opaadv i ⟨⟨i qmsg ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))⟩⟩"
proof -
have "opaadv i ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
  by (rule ostep_invariant_weakenE [OF olocal_quality_increases])
  (auto dest!: onllD seqllD elim!: aadv_ex_labelE)
hence "opaadv i ⟨⟨i qmsg ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))⟩⟩"
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

lemma par_rreq_rrep_nsqn_fresh_any_step:
shows "opaadv i ⟨⟨i qmsg ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ,
      other (λ_ _. True) {i} →)
      globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)⟩⟩"
proof -
have "opaadv i ⊢A (λσ _. (orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →)
      globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)"
proof (rule ostep_invariant_weakenE [OF rreq_rrep_nsqn_fresh_any_step_invariant])
fix t
assume "onll ΓAODV (λ((σ, _), a, _). anycast (msg_fresh σ) a) t"
thus "globala (λ(σ, a, σ'). anycast (msg_fresh σ) a) t"
  by (cases t) (clarsimp dest!: onllD, metis aadv_ex_label)
qed auto
hence "opaadv i ⟨⟨i qmsg ⊢A (λσ _. (orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →)
      globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)⟩⟩"
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

lemma par_anycast_msg_zhops:
shows "opaadv i ⟨⟨i qmsg ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(_, a, _). anycast msg_zhops a)⟩⟩"
proof -
from anycast_msg_zhops initiali_aadv oaadv_trans aadv_trans
have "opaadv i ⊢A (act TT, other (λ_ _. True) {i} →)
      seqll i (onll ΓAODV (λ(_, a, _). anycast msg_zhops a))"
  by (rule open_seq_step_invariant)
hence "opaadv i ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(_, a, _). anycast msg_zhops a)"
proof (rule ostep_invariant_weakenE)
fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
assume "seqll i (onll ΓAODV (λ(_, a, _). anycast msg_zhops a)) t"
thus "globala (λ(_, a, _). anycast msg_zhops a) t"
  by (cases t) (clarsimp dest!: seqllD onllD, metis aadv_ex_label)
qed simp_all
hence "opaadv i ⟨⟨i qmsg ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(_, a, _). anycast msg_zhops a)⟩⟩"

```

```

  by (rule lift_step_into_qmsg_statelessassm) simp_all
  thus ?thesis by rule auto
qed

```

## 0.12.2 Lift to nodes

lemma node\_step\_no\_change\_on\_send\_or\_receive:

```

  assumes "(( $\sigma$ , NodeS i P R), a, ( $\sigma'$ , NodeS i' P' R'))  $\in$  onode_sos
          (oparp_sos i (oseqp_sos  $\Gamma_{AODV}$  i) (seqp_sos  $\Gamma_{QMSG}$ ))"
  and "a  $\neq \tau$ "
  shows " $\sigma' i = \sigma i$ "
  using assms
  by (cases a) (auto elim!: par_step_no_change_on_send_or_receive)

```

lemma node\_nhop\_quality\_increases:

```

  shows " $\langle i : opaadv i \langle \langle i \text{ qmsg} : R \rangle_o \models$ 
        (otherwith (op=) {i}
          (oarrivmsg ( $\lambda \sigma m.$  msg_fresh  $\sigma m \wedge$  msg_zhops m)),
          other quality_increases {i}
         $\rightarrow$  global ( $\lambda \sigma.$   $\forall$  dip. let nhip = the (nhop (rt ( $\sigma i$ )) dip)
          in dip  $\in$  vD (rt ( $\sigma i$ ))  $\cap$  vD (rt ( $\sigma$  nhip))  $\wedge$  nhip  $\neq$  dip
           $\rightarrow$  (rt ( $\sigma i$ ))  $\sqsubseteq_{dip}$  (rt ( $\sigma$  nhip)))")"
  by (rule node_lift [OF par_nhop_quality_increases]) auto

```

lemma node\_quality\_increases:

```

  " $\langle i : opaadv i \langle \langle i \text{ qmsg} : R \rangle_o \models_A$  ( $\lambda \sigma _.$  oarrivmsg ( $\lambda _.$  rreq_rrep_sn)  $\sigma$ ,
        other ( $\lambda _.$  True) {i}  $\rightarrow$ )
        globala ( $\lambda (\sigma, _, \sigma').$  quality_increases ( $\sigma i$ ) ( $\sigma' i$ ))"
  by (rule node_lift_step_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp

```

lemma node\_rreq\_rrep\_nsqn\_fresh\_any\_step:

```

  shows " $\langle i : opaadv i \langle \langle i \text{ qmsg} : R \rangle_o \models_A$ 
        ( $\lambda \sigma _.$  oarrivmsg ( $\lambda _.$  rreq_rrep_sn)  $\sigma$ , other ( $\lambda _.$  True) {i}  $\rightarrow$ )
        globala ( $\lambda (\sigma, a, \sigma').$  castmsg (msg_fresh  $\sigma$ ) a)"
  by (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_nsqn_fresh_any_step])

```

lemma node\_anycast\_msg\_zhops:

```

  shows " $\langle i : opaadv i \langle \langle i \text{ qmsg} : R \rangle_o \models_A$ 
        ( $\lambda \sigma _.$  oarrivmsg ( $\lambda _.$  rreq_rrep_sn)  $\sigma$ , other ( $\lambda _.$  True) {i}  $\rightarrow$ )
        globala ( $\lambda (_, a, _).$  castmsg msg_zhops a)"
  by (rule node_lift_anycast_statelessassm [OF par_anycast_msg_zhops])

```

lemma node\_silent\_change\_only:

```

  shows " $\langle i : opaadv i \langle \langle i \text{ qmsg} : R_i \rangle_o \models_A$  ( $\lambda \sigma _.$  oarrivmsg ( $\lambda _.$  True)  $\sigma$ ,
        other ( $\lambda _.$  True) {i}  $\rightarrow$ )
        globala ( $\lambda (\sigma, a, \sigma').$  a  $\neq \tau \rightarrow \sigma' i = \sigma i$ )"

```

proof (rule ostep\_invariantI, simp (no\_asm), rule impI)

```

  fix  $\sigma \zeta a \sigma' \zeta'$ 
  assume or: "(( $\sigma, \zeta$ )  $\in$  oreachable ( $\langle i : opaadv i \langle \langle i \text{ qmsg} : R_i \rangle_o$ )
        ( $\lambda \sigma _.$  oarrivmsg ( $\lambda _.$  True)  $\sigma$ )
        (other ( $\lambda _.$  True) {i})))"
  and tr: "(( $\sigma, \zeta$ ), a, ( $\sigma', \zeta'$ ))  $\in$  trans ( $\langle i : opaadv i \langle \langle i \text{ qmsg} : R_i \rangle_o$ )"
  and "a  $\neq \tau_n$ "
  from or obtain p R where " $\zeta =$  NodeS i p R"
  by - (drule node_net_state, metis)
  with tr have "(( $\sigma, \text{NodeS i p R}$ ), a, ( $\sigma', \zeta'$ ))
         $\in$  onode_sos (oparp_sos i (trans (opaadv i)) (trans qmsg))"
  by simp
  thus " $\sigma' i = \sigma i$ " using <a  $\neq \tau_n$ >
  by (cases rule: onode_sos.cases)
    (auto elim: qmsg_no_change_on_send_or_receive)

```

qed

### 0.12.3 Lift to partial networks

```

lemma arrive_rreq_rrep_nsqn_fresh_inc_sn [simp]:
  assumes "oarrivemsg ( $\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge P \sigma m$ )  $\sigma m$ "
  shows "oarrivemsg ( $\lambda_. \text{rreq\_rrep\_sn}$ )  $\sigma m$ "
  using assms by (cases m) auto

lemma opnet_nhop_quality_increases:
  shows "opnet ( $\lambda i. \text{opaodv } i \langle\langle i \text{ qmsg} \rangle\rangle p \models$ 
    (otherwith (op=) (net_tree_ips p)
      (oarrivemsg ( $\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m$ )),
      other quality_increases (net_tree_ips p)  $\rightarrow$ )
    global ( $\lambda\sigma. \forall i \in \text{net\_tree\_ips } p. \forall \text{dip.}$ 
      let nhip = the (nhop (rt ( $\sigma i$ )) dip)
      in dip  $\in vD$  (rt ( $\sigma i$ ))  $\cap vD$  (rt ( $\sigma \text{nhip}$ ))  $\wedge$  nhip  $\neq$  dip
       $\rightarrow$  (rt ( $\sigma i$ ))  $\sqsubset_{\text{dip}}$  (rt ( $\sigma \text{nhip}$ )))")"

proof (rule pnet_lift [OF node_nhop_quality_increases])
  fix i R
  have " $\langle i : \text{opaodv } i \langle\langle i \text{ qmsg} : R \rangle\rangle_o \models_A (\lambda\sigma \_ . \text{oarrivemsg } (\lambda_. \text{rreq\_rrep\_sn}) \sigma,$ 
    other ( $\lambda\_ \_ . \text{True}$ ) {i}  $\rightarrow$ ) globala ( $\lambda(\sigma, a, \sigma')$ ).
    castmsg ( $\lambda m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m$ ) a)"
  proof (rule ostep_invariantI, simp (no_asm))
    fix  $\sigma s a \sigma' s'$ 
    assume or: " $(\sigma, s) \in \text{oreachable } (\langle i : \text{opaodv } i \langle\langle i \text{ qmsg} : R \rangle\rangle_o$ 
      ( $\lambda\sigma \_ . \text{oarrivemsg } (\lambda_. \text{rreq\_rrep\_sn}) \sigma$ )
      (other ( $\lambda\_ \_ . \text{True}$ ) {i}))"
      and tr: " $((\sigma, s), a, (\sigma', s')) \in \text{trans } (\langle i : \text{opaodv } i \langle\langle i \text{ qmsg} : R \rangle\rangle_o)$ "
      and am: " $\text{oarrivemsg } (\lambda_. \text{rreq\_rrep\_sn}) \sigma a$ "
    from or tr am have "castmsg (msg_fresh  $\sigma$ ) a"
    by (auto dest!: ostep_invariantD [OF node_rreq_rrep_nsqn_fresh_any_step])
    moreover from or tr am have "castmsg (msg_zhops) a"
    by (auto dest!: ostep_invariantD [OF node_anycast_msg_zhops])
    ultimately show "castmsg ( $\lambda m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m$ ) a"
    by (case_tac a) auto
  qed
  thus " $\langle i : \text{opaodv } i \langle\langle i \text{ qmsg} : R \rangle\rangle_o \models_A$ 
    ( $\lambda\sigma \_ . \text{oarrivemsg } (\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) \sigma,$ 
    other quality_increases {i}  $\rightarrow$ ) globala ( $\lambda(\sigma, a, \_)$ .
    castmsg ( $\lambda m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m$ ) a)"
  by rule auto
next
  fix i R
  show " $\langle i : \text{opaodv } i \langle\langle i \text{ qmsg} : R \rangle\rangle_o \models_A$ 
    ( $\lambda\sigma \_ . \text{oarrivemsg } (\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) \sigma,$ 
    other quality_increases {i}  $\rightarrow$ ) globala ( $\lambda(\sigma, a, \sigma')$ .
    a  $\neq \tau \wedge (\forall i d. a \neq i:\text{deliver}(d)) \rightarrow \sigma i = \sigma' i$ )"
  by (rule ostep_invariant_weakenE [OF node_silent_change_only]) auto
next
  fix i R
  show " $\langle i : \text{opaodv } i \langle\langle i \text{ qmsg} : R \rangle\rangle_o \models_A$ 
    ( $\lambda\sigma \_ . \text{oarrivemsg } (\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) \sigma,$ 
    other quality_increases {i}  $\rightarrow$ ) globala ( $\lambda(\sigma, a, \sigma')$ .
    a =  $\tau \vee (\exists d. a = i:\text{deliver}(d)) \rightarrow \text{quality\_increases } (\sigma i) (\sigma' i)$ )"
  by (rule ostep_invariant_weakenE [OF node_quality_increases]) auto
qed simp_all

```

### 0.12.4 Lift to closed networks

```

lemma onet_nhop_quality_increases:
  shows "oclosed (opnet ( $\lambda i. \text{opaodv } i \langle\langle i \text{ qmsg} \rangle\rangle p$ )
     $\models (\lambda\_ \_ \_ . \text{True}, \text{other quality\_increases } (\text{net\_tree\_ips } p) \rightarrow)$ 
    global ( $\lambda\sigma. \forall i \in \text{net\_tree\_ips } p. \forall \text{dip.}$ 
      let nhip = the (nhop (rt ( $\sigma i$ )) dip)
      in dip  $\in vD$  (rt ( $\sigma i$ ))  $\cap vD$  (rt ( $\sigma \text{nhip}$ ))  $\wedge$  nhip  $\neq$  dip
       $\rightarrow$  (rt ( $\sigma i$ ))  $\sqsubset_{\text{dip}}$  (rt ( $\sigma \text{nhip}$ )))")"

```

```

(is "_  $\models$  (_, ?U  $\rightarrow$ ) ?inv")
proof (rule inclosed_closed)
  from opnet_nhop_quality_increases
  show "opnet ( $\lambda i$ . opaadv i  $\langle\langle i$  qmsg) p
     $\models$  (otherwith (op=) (net_tree_ips p) inclosed, ?U  $\rightarrow$ ) ?inv"
proof (rule oinvariant_weakenE)
  fix  $\sigma \sigma' ::$  "ip  $\Rightarrow$  state" and a :: "msg node_action"
  assume "otherwith (op=) (net_tree_ips p) inclosed  $\sigma \sigma'$  a"
  thus "otherwith (op=) (net_tree_ips p)
    (oarrivemsg ( $\lambda \sigma m$ . msg_fresh  $\sigma m \wedge$  msg_zhops m))  $\sigma \sigma'$  a"
proof (rule otherwithEI)
  fix  $\sigma ::$  "ip  $\Rightarrow$  state" and a :: "msg node_action"
  assume "inclosed  $\sigma$  a"
  thus "oarrivemsg ( $\lambda \sigma m$ . msg_fresh  $\sigma m \wedge$  msg_zhops m)  $\sigma$  a"
  proof (cases a)
    fix ii ni ms
    assume "a = ii $\neg$ ni:arrive(ms)"
    moreover with (inclosed  $\sigma$  a) obtain d di where "ms = newpkt(d, di)"
    by (cases ms) auto
    ultimately show ?thesis by simp
  qed simp_all
qed
qed
qed
qed

```

### 0.12.5 Transfer into the standard model

```

interpretation aadv_openproc: openproc paadv opaadv id
  where "aadv_openproc.initmissing = initmissing"
proof -
  show "openproc paadv opaadv id"
  proof unfold_locales
    fix i :: ip
    have "{( $\sigma$ ,  $\zeta$ ). ( $\sigma$  i,  $\zeta$ )  $\in$   $\sigma_{AODV}$  i  $\wedge$  ( $\forall j$ .  $j \neq i \rightarrow \sigma j \in$  fst '  $\sigma_{AODV}$  j)}  $\subseteq$   $\sigma_{AODV}'$ "
      unfolding  $\sigma_{AODV\_def}$   $\sigma_{AODV}'\_def$ 
      proof (rule equalityD1)
        show " $\bigwedge f p$ . {( $\sigma$ ,  $\zeta$ ). ( $\sigma$  i,  $\zeta$ )  $\in$  {(f i, p)}  $\wedge$  ( $\forall j$ .  $j \neq i$ 
           $\rightarrow \sigma j \in$  fst ' {(f j, p)})} = {(f, p)}"
          by (rule set_eqI) auto
      qed
    thus "{( $\sigma$ ,  $\zeta$ ) |  $\sigma \zeta s$ .  $s \in$  init (paadv i)
       $\wedge$  ( $\sigma$  i,  $\zeta$ ) = id s
       $\wedge$  ( $\forall j$ .  $j \neq i \rightarrow \sigma j \in$  (fst o id) ' init (paadv j)) }  $\subseteq$  init (opaadv i)"
      by simp
  next
    show " $\forall j$ . init (paadv j)  $\neq$  {}"
      unfolding  $\sigma_{AODV\_def}$  by simp
  next
    fix i s a s'  $\sigma \sigma'$ 
    assume " $\sigma$  i = fst (id s)"
      and " $\sigma'$  i = fst (id s)"
      and "(s, a, s')  $\in$  trans (paadv i)"
    then obtain q q' where "s = ( $\sigma$  i, q)"
      and "s' = ( $\sigma'$  i, q)"
      and "(( $\sigma$  i, q), a, ( $\sigma'$  i, q'))  $\in$  trans (paadv i)"
      by (cases s, cases s') auto
    from this(3) have "(( $\sigma$ , q), a, ( $\sigma'$ , q'))  $\in$  trans (opaadv i)"
      by simp (rule open_seqp_action [OF aadv_wf])

    with  $\langle s = (\sigma$  i, q)  $\rangle$  and  $\langle s' = (\sigma'$  i, q')  $\rangle$ 
      show "(( $\sigma$ , snd (id s)), a, ( $\sigma'$ , snd (id s')))  $\in$  trans (opaadv i)"
      by simp
  qed
then interpret op: openproc paadv opaadv id .

```



```

have [simp]: "∧i. (SOME x. x ∈ (fst o id) ' init (paodv i)) = aadv_init i"
  unfolding  $\sigma_{AODV\_def}$  by simp
hence "∧i. openproc.initmissing paodv id i = initmissing i"
  unfolding op.initmissing_def op.someinit_def initmissing_def
  by (auto split: option.split)
thus "openproc.initmissing paodv id = initmissing" ..
qed

```

interpretation aadv\_openproc\_par\_qmsg: openproc\_parq paodv opaodv id qmsg

```

where "aadv_openproc_par_qmsg.netglobal = netglobal"
  and "aadv_openproc_par_qmsg.initmissing = initmissing"

```

proof -

```

show "openproc_parq paodv opaodv id qmsg"
  by (unfold_locales) simp
then interpret opq: openproc_parq paodv opaodv id qmsg .

```

```

have im: "∧ $\sigma$ . openproc.initmissing ( $\lambda$ i. paodv i << qmsg) ( $\lambda$ (p, q). (fst (id p), snd (id p), q))  $\sigma$ 
  = initmissing  $\sigma$ "

```

```

  unfolding opq.initmissing_def opq.someinit_def initmissing_def
  unfolding  $\sigma_{AODV\_def}$   $\sigma_{QMSG\_def}$  by (clarsimp cong: option.case_cong)

```

```

thus "openproc.initmissing ( $\lambda$ i. paodv i << qmsg) ( $\lambda$ (p, q). (fst (id p), snd (id p), q)) = initmissing"
  by (rule ext)

```

```

have "∧P  $\sigma$ . openproc.netglobal ( $\lambda$ i. paodv i << qmsg) ( $\lambda$ (p, q). (fst (id p), snd (id p), q)) P  $\sigma$ 
  = netglobal P  $\sigma$ "

```

```

  unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def
  unfolding  $\sigma_{AODV\_def}$   $\sigma_{QMSG\_def}$ 

```

```

  by (clarsimp cong: option.case_cong
      simp del: One_nat_def
      simp add: fst_initmissing_netgmap_default_aadv_init_netlift
      [symmetric, unfolded initmissing_def])

```

```

thus "openproc.netglobal ( $\lambda$ i. paodv i << qmsg) ( $\lambda$ (p, q). (fst (id p), snd (id p), q)) = netglobal"
  by auto

```

qed

lemma net\_nhop\_quality\_increases:

```

assumes "wf_net_tree n"

```

```

shows "closed (pnet ( $\lambda$ i. paodv i << qmsg) n)  $\models$  netglobal
  ( $\lambda$  $\sigma$ .  $\forall$ i dip. let nhip = the (nhop (rt ( $\sigma$  i)) dip)
    in dip  $\in$  vD (rt ( $\sigma$  i))  $\cap$  vD (rt ( $\sigma$  nhip))  $\wedge$  nhip  $\neq$  dip
     $\rightarrow$  (rt ( $\sigma$  i))  $\sqsubset_{dip}$  (rt ( $\sigma$  nhip)))"
  (is "_  $\models$  netglobal ( $\lambda$  $\sigma$ .  $\forall$ i. ?inv  $\sigma$  i)")

```

proof -

```

from (wf_net_tree n)

```

```

  have proto: "closed (pnet ( $\lambda$ i. paodv i << qmsg) n)  $\models$  netglobal ( $\lambda$  $\sigma$ .  $\forall$ i $\in$ net_tree_ips n.  $\forall$ dip.
    let nhip = the (nhop (rt ( $\sigma$  i)) dip)
    in dip  $\in$  vD (rt ( $\sigma$  i))  $\cap$  vD (rt ( $\sigma$  nhip))  $\wedge$  nhip  $\neq$  dip
     $\rightarrow$  (rt ( $\sigma$  i))  $\sqsubset_{dip}$  (rt ( $\sigma$  nhip)))"

```

```

  by (rule aadv_openproc_par_qmsg.close_opnet [OF _ onet_nhop_quality_increases])

```

show ?thesis

```

unfolding invariant_def opnet_sos.opnet_tau1

```

```

proof (rule, simp only: aadv_openproc_par_qmsg.netglobalsimp
  fst_initmissing_netgmap_pair_fst, rule allI)

```

```

  fix  $\sigma$  i

```

```

  assume sr: " $\sigma \in$  reachable (closed (pnet ( $\lambda$ i. paodv i << qmsg) n)) TT"

```

```

  hence " $\forall$ i $\in$ net_tree_ips n. ?inv (fst (initmissing (netgmap fst  $\sigma$ ))) i"

```

```

    by - (drule invariantD [OF proto],
      simp only: aadv_openproc_par_qmsg.netglobalsimp
      fst_initmissing_netgmap_pair_fst)

```

```

  thus "?inv (fst (initmissing (netgmap fst  $\sigma$ ))) i"

```

```

  proof (cases "i $\in$ net_tree_ips n")

```

```

    assume "i $\notin$ net_tree_ips n"

```

```

    from sr have " $\sigma \in$  reachable (pnet ( $\lambda$ i. paodv i << qmsg) n) TT" ..

```

```

    hence "net_ips  $\sigma$  = net_tree_ips n" ..

```

```

    with (i $\notin$ net_tree_ips n) have "i $\notin$ net_ips  $\sigma$ " by simp

```

```

    hence "(fst (initmissing (netgmap fst  $\sigma$ ))) i = aadv_init i"
      by simp
    thus ?thesis by simp
  qed metis
qed
qed

```

### 0.12.6 Loop freedom of AODV

**theorem** *aadv\_loop\_freedom*:

assumes "*wf\_net\_tree* *n*"

shows "*closed* (*pnet* ( $\lambda i$ . *paadv* *i*  $\langle\langle$  *qmsg* *n*  $\rangle\rangle$ )  $\models$  *netglobal* ( $\lambda\sigma$ .  $\forall dip$ . *irrefl* (*rt\_graph*  $\sigma$  *dip*)<sup>+</sup>)"

using *assms* by (rule *aadv\_openproc\_par\_qmsg.netglobal\_weakenE*  
                   [*OF net\_nhop\_quality\_increases\_inv\_to\_loop\_freedom*])

**end**

# Chapter 1

## Variant A: Skipping the RREQ ID

Explanation [4, §10.1]: AODV does not need the route request identifier. This number, in combination with the IP address of the originator, is used to identify every RREQ message in a unique way. This variant shows that the combination of the originators IP address and its sequence number is just as suited to uniquely determine the route request to which the message belongs. Hence, the route request identifier field is not required. This can then reduce the size of the RREQ message.

### 1.1 Predicates and functions used in the AODV model

```
theory A_Aodv_Data
imports A_Norreqid
begin
```

#### 1.1.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

```
definition inc :: "sqn  $\Rightarrow$  sqn"
  where "inc sn  $\equiv$  if sn = 0 then sn else sn + 1"
```

```
lemma less_than_inc [simp]: "x  $\leq$  inc x"
  unfolding inc_def by simp
```

```
lemma inc_minus_suc_0 [simp]:
  "inc x - Suc 0 = x"
  unfolding inc_def by simp
```

```
lemma inc_never_one' [simp, intro]: "inc x  $\neq$  Suc 0"
  unfolding inc_def by simp
```

```
lemma inc_never_one [simp, intro]: "inc x  $\neq$  1"
  by simp
```

#### 1.1.2 Modelling Routes

A route is a 6-tuple,  $(dsn, dsk, flag, hops, nhip, pre)$  where  $dsn$  is the ‘destination sequence number’,  $dsk$  is the ‘destination-sequence-number status’,  $flag$  is the route status,  $hops$  is the number of hops to the destination,  $nhip$  is the next hop toward the destination, and  $pre$  is the set of ‘precursor nodes’ those interested in hearing about changes to the route.

```
type_synonym r = "sqn  $\times$  k  $\times$  f  $\times$  nat  $\times$  ip  $\times$  ip set"
```

```
definition proj2 :: "r  $\Rightarrow$  sqn" ("π2")
  where "π2  $\equiv$   $\lambda$ (dsn, _, _, _, _). dsn"
```

```
definition proj3 :: "r  $\Rightarrow$  k" ("π3")
  where "π3  $\equiv$   $\lambda$ (_, dsk, _, _, _). dsk"
```

```
definition proj4 :: "r  $\Rightarrow$  f" ("π4")
```

```

where "π4 ≡ λ(⟦_, _⟧, flag, _, _, _). flag"

definition proj5 :: "r ⇒ nat" ("π5")
  where "π5 ≡ λ(⟦_, _⟧, _, hops, _, _). hops"

definition proj6 :: "r ⇒ ip" ("π6")
  where "π6 ≡ λ(⟦_, _⟧, _, _, nhip, _). nhip"

definition proj7 :: "r ⇒ ip set" ("π7")
  where "π7 ≡ λ(⟦_, _⟧, _, _, _, pre). pre"

lemma projs [simp]:
  "π2(dsn, dsk, flag, hops, nhip, pre) = dsn"
  "π3(dsn, dsk, flag, hops, nhip, pre) = dsk"
  "π4(dsn, dsk, flag, hops, nhip, pre) = flag"
  "π5(dsn, dsk, flag, hops, nhip, pre) = hops"
  "π6(dsn, dsk, flag, hops, nhip, pre) = nhip"
  "π7(dsn, dsk, flag, hops, nhip, pre) = pre"
  by (clarsimp simp: proj2_def proj3_def proj4_def
      proj5_def proj6_def proj7_def)+

lemma proj3_pred [intro]: "⟦ P kno; P unk ⟧ ⇒ P (π3 x)"
  by (rule k.induct)

lemma proj4_pred [intro]: "⟦ P val; P inv ⟧ ⇒ P (π4 x)"
  by (rule f.induct)

lemma proj6_pair_snd [simp]:
  fixes dsn' r
  shows "π6 (dsn', snd (r)) = π6(r)"
  by (cases r) simp



### 1.1.3 Routing Tables



Routing tables map ip addresses to route entries.



```

type_synonym rt = "ip → r"

syntax
  "_Sigma_route" :: "rt ⇒ ip → r" ("σroute'(⟦_, _⟧)")

translations
  "σroute(rt, dip)" => "rt dip"

definition sqn :: "rt ⇒ ip ⇒ sqn"
  where "sqn rt dip ≡ case σroute(rt, dip) of Some r ⇒ π2(r) | None ⇒ 0"

definition sqnf :: "rt ⇒ ip ⇒ k"
  where "sqnf rt dip ≡ case σroute(rt, dip) of Some r ⇒ π3(r) | None ⇒ unk"

abbreviation flag :: "rt ⇒ ip → f"
  where "flag rt dip ≡ map_option π4 (σroute(rt, dip))"

abbreviation dhops :: "rt ⇒ ip → nat"
  where "dhops rt dip ≡ map_option π5 (σroute(rt, dip))"

abbreviation nhop :: "rt ⇒ ip → ip"
  where "nhop rt dip ≡ map_option π6 (σroute(rt, dip))"

abbreviation precs :: "rt ⇒ ip → ip set"
  where "precs rt dip ≡ map_option π7 (σroute(rt, dip))"

definition vD :: "rt ⇒ ip set"
  where "vD rt ≡ {dip. flag rt dip = Some val}"

```


```

```

definition iD :: "rt ⇒ ip set"
  where "iD rt ≡ {dip. flag rt dip = Some inv}"

definition kD :: "rt ⇒ ip set"
  where "kD rt ≡ {dip. rt dip ≠ None}"

lemma kD_is_vD_and_iD: "kD rt = vD rt ∪ iD rt"
  unfolding kD_def vD_def iD_def by auto

lemma vD_iD_gives_kD [simp]:
  "∧ip rt. ip ∈ vD rt ⇒ ip ∈ kD rt"
  "∧ip rt. ip ∈ iD rt ⇒ ip ∈ kD rt"
  unfolding kD_is_vD_and_iD by simp_all

lemma kD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ kD rt"
  shows "∃ dsn dsk flag hops nhip pre.
    σroute(rt, dip) = Some (dsn, dsk, flag, hops, nhip, pre)"
  using assms unfolding kD_def by simp

lemma kD_None [dest]:
  fixes dip rt
  assumes "dip ∉ kD rt"
  shows "σroute(rt, dip) = None"
  using assms unfolding kD_def
  by (metis (mono_tags) mem_Collect_eq)

lemma vD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ vD rt"
  shows "∃ dsn dsk hops nhip pre.
    σroute(rt, dip) = Some (dsn, dsk, val, hops, nhip, pre)"
  using assms unfolding vD_def by simp

lemma vD_empty [simp]: "vD Map.empty = {}"
  unfolding vD_def by simp

lemma iD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ iD rt"
  shows "∃ dsn dsk hops nhip pre.
    σroute(rt, dip) = Some (dsn, dsk, inv, hops, nhip, pre)"
  using assms unfolding iD_def by simp

lemma val_is_vD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "the (flag rt ip) = val"
  shows "ip ∈ vD(rt)"
  using assms unfolding vD_def by auto

lemma inv_is_iD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "the (flag rt ip) = inv"
  shows "ip ∈ iD(rt)"
  using assms unfolding iD_def by auto

lemma iD_flag_is_inv [elim, simp]:
  fixes ip rt
  assumes "ip ∈ iD(rt)"
  shows "the (flag rt ip) = inv"
  proof -

```

```

    from ⟨ip∈iD(rt)⟩ have "ip∈kD(rt)" by auto
    with assms show ?thesis unfolding iD_def by auto
qed

lemma kD_but_not_vD_is_iD [elim]:
  fixes ip rt
  assumes "ip∈kD(rt)"
    and "ip∉vD(rt)"
  shows "ip∈iD(rt)"
proof -
  from ⟨ip∈kD(rt)⟩ obtain dsn dsk f hops nhop pre
    where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop, pre)"
    by (metis kD_Some)
  from ⟨ip∉vD(rt)⟩ have "f ≠ val"
  proof (rule contrapos_nn)
    assume "f = val"
    with rtip have "the (flag rt ip) = val" by simp
    with ⟨ip∈kD(rt)⟩ show "ip∈vD(rt)" ..
  qed
  with rtip have "the (flag rt ip) = inv" by simp
  with ⟨ip∈kD(rt)⟩ show "ip∈iD(rt)" ..
qed

lemma vD_or_iD [elim]:
  fixes ip rt
  assumes "ip∈kD(rt)"
    and "ip∈vD(rt) ⇒ P rt ip"
    and "ip∈iD(rt) ⇒ P rt ip"
  shows "P rt ip"
proof -
  from ⟨ip∈kD(rt)⟩ have "ip∈vD(rt) ∪ iD(rt)"
    by (simp add: kD_is_vD_and_iD)
  thus ?thesis by (auto elim: assms(2-3))
qed

lemma proj5_eq_dhops: "∧dip rt. dip∈kD(rt) ⇒ π5(the (rt dip)) = the (dhops rt dip)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj4_eq_flag: "∧dip rt. dip∈kD(rt) ⇒ π4(the (rt dip)) = the (flag rt dip)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj2_eq_sqn: "∧dip rt. dip∈kD(rt) ⇒ π2(the (rt dip)) = sqn rt dip"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 [simp]:
  "∧ip rt. ip∈kD(rt) ⇒ sqnf rt ip = π3(the (rt ip))"
  unfolding sqnf_def by auto

lemma vD_flag_val [simp]:
  "∧dip rt. dip ∈ vD (rt) ⇒ the (flag rt dip) = val"
  unfolding vD_def by clarsimp

lemma kD_update [simp]:
  "∧rt nip v. kD (rt(nip ↦ v)) = insert nip (kD rt)"
  unfolding kD_def by auto

lemma kD_empty [simp]: "kD Map.empty = {}"
  unfolding kD_def by simp

lemma ip_equal_or_known [elim]:
  fixes rt ip ip'
  assumes "ip = ip' ∨ ip∈kD(rt)"
    and "ip = ip' ⇒ P rt ip ip'"
    and "[ ip ≠ ip'; ip∈kD(rt) ] ⇒ P rt ip ip'"

```

```

shows "P rt ip ip'"
using assms by auto

```

### 1.1.4 Updating Routing Tables

Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.

#### Updating Precursor Lists

```

definition addpre :: "r ⇒ ip set ⇒ r"
  where "addpre r npre ≡ let (dsn, dsk, flag, hops, nhip, pre) = r in
        (dsn, dsk, flag, hops, nhip, pre ∪ npre)"

```

```

lemma proj2_addpre:
  fixes v pre
  shows "π2(addpre v pre) = π2(v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj3_addpre:
  fixes v pre
  shows "π3(addpre v pre) = π3(v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj4_addpre:
  fixes v pre
  shows "π4(addpre v pre) = π4(v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj5_addpre:
  fixes v pre
  shows "π5(addpre v pre) = π5(v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj6_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows "π6(addpre v npre) = π6(v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj7_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows "π7(addpre v npre) = π7(v) ∪ npre"
  unfolding addpre_def by (cases v) simp

```

```

lemma addpre_empty: "addpre r {} = r"
  unfolding addpre_def by simp

```

```

lemma addpre_r:
  "addpre (dsn, dsk, fl, hops, nhip, pre) npre = (dsn, dsk, fl, hops, nhip, pre ∪ npre)"
  unfolding addpre_def by simp

```

```

lemmas addpre_simps [simp] = proj2_addpre proj3_addpre proj4_addpre proj5_addpre
  proj6_addpre proj7_addpre addpre_empty addpre_r

```

```

definition addpreRT :: "rt ⇒ ip ⇒ ip set → rt"
  where "addpreRT rt dip npre ≡
        map_option (λs. rt (dip ↦ addpre s npre)) (σroute(rt, dip))"

```

```

lemma snd_addpre [simp]:
  "∧dsn dsn' v pre. (dsn, snd(addpre (dsn', v) pre)) = addpre (dsn, v) pre"
  unfolding addpre_def by clarsimp

```

```

lemma proj2_addpreRT [simp]:
  fixes ip rt ip' npre

```

```

assumes "ip ∈ kD rt"
  and "ip' ∈ kD rt"
  shows "π2(the (the (addpreRT rt ip' npre) ip)) = π2(the (rt ip))"
using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj3_addpreRT [simp]:
  fixes ip rt ip' npre
  assumes "ip ∈ kD rt"
    and "ip' ∈ kD rt"
  shows "π3(the (the (addpreRT rt ip' npre) ip)) = π3(the (rt ip))"
using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj5_addpreRT [simp]:
  "∧rt dip ip npre. dip ∈ kD(rt) ⇒ π5(the (the (addpreRT rt dip npre) ip)) = π5(the (rt ip))"
unfolding addpreRT_def by auto

lemma flag_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "flag (the (addpreRT rt dip pre)) ip = flag rt ip"
unfolding addpreRT_def
using assms [THEN kD_Some] by (clarsimp)

lemma kD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip ∈ kD rt"
  shows "kD (the (addpreRT rt dip npre)) = kD rt"
unfolding kD_def addpreRT_def
using assms [THEN kD_Some]
by clarsimp blast

lemma vD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip ∈ kD rt"
  shows "vD (the (addpreRT rt dip npre)) = vD rt"
unfolding vD_def addpreRT_def
using assms [THEN kD_Some] by clarsimp auto

lemma iD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip ∈ kD rt"
  shows "iD (the (addpreRT rt dip npre)) = iD rt"
unfolding iD_def addpreRT_def
using assms [THEN kD_Some] by clarsimp auto

lemma nhop_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "nhop (the (addpreRT rt dip pre)) ip = nhop rt ip"
unfolding sqn_def addpreRT_def
using assms [THEN kD_Some] by (clarsimp)

lemma sqn_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "sqn (the (addpreRT rt dip pre)) ip = sqn rt ip"
unfolding sqn_def addpreRT_def
using assms [THEN kD_Some] by (clarsimp)

lemma dhops_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "dhops (the (addpreRT rt dip pre)) ip = dhops rt ip"
unfolding addpreRT_def

```



```

using assms [THEN kD_Some] by (clarsimp)

lemma sqnf_addpreRT [simp]:
  " $\bigwedge ip\ dip. ip \in kD(rt\ \xi) \implies sqnf\ (the\ (addpreRT\ (rt\ \xi)\ ip\ npre))\ dip = sqnf\ (rt\ \xi)\ dip$ "
  unfolding sqnf_def addpreRT_def by auto

Updating route entries

lemma in_kD_case [simp]:
  fixes dip rt
  assumes "dip  $\in$  kD(rt)"
  shows "(case rt dip of None  $\implies$  en | Some r  $\implies$  es r) = es (the (rt dip))"
  using assms [THEN kD_Some] by auto

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip  $\notin$  kD(rt)"
  shows "(case rt dip of None  $\implies$  en | Some r  $\implies$  es r) = en"
  using assms [THEN kD_None] by auto

lemma rt_Some_sqn [dest]:
  fixes rt and ip dsn dsk flag hops nhip pre
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows "sqn rt ip = dsn"
  unfolding sqn_def using assms by simp

lemma not_kD_sqn [simp]:
  fixes dip rt
  assumes "dip  $\notin$  kD(rt)"
  shows "sqn rt dip = 0"
  using assms unfolding sqn_def
  by simp

definition update_arg_wf :: "r  $\implies$  bool"
where "update_arg_wf r  $\equiv$   $\pi_4(r) = val \wedge$ 
      ( $\pi_2(r) = 0$ ) = ( $\pi_3(r) = unk$ )  $\wedge$ 
      ( $\pi_3(r) = unk \implies \pi_5(r) = 1$ )"

lemma update_arg_wf_gives_cases:
  " $\bigwedge r. update\_arg\_wf\ r \implies (\pi_2(r) = 0) = (\pi_3(r) = unk)$ "
  unfolding update_arg_wf_def by simp

lemma update_arg_wf_tuples [simp]:
  " $\bigwedge nhip\ pre. update\_arg\_wf\ (0, unk, val, Suc\ 0, nhip, pre)$ "
  " $\bigwedge n\ hops\ nhip\ pre. update\_arg\_wf\ (Suc\ n, kno, val, hops, nhip, pre)$ "
  unfolding update_arg_wf_def by auto

lemma update_arg_wf_tuples' [elim]:
  " $\bigwedge n\ hops\ nhip\ pre. Suc\ 0 \leq n \implies update\_arg\_wf\ (n, kno, val, hops, nhip, pre)$ "
  unfolding update_arg_wf_def by auto

lemma wf_r_cases [intro]:
  fixes P r
  assumes "update_arg_wf r"
  and c1: " $\bigwedge nhip\ pre. P\ (0, unk, val, Suc\ 0, nhip, pre)$ "
  and c2: " $\bigwedge dsn\ hops\ nhip\ pre. dsn > 0 \implies P\ (dsn, kno, val, hops, nhip, pre)$ "
  shows "P r"
proof -
  obtain dsn dsk flag hops nhip pre
  where *: "r = (dsn, dsk, flag, hops, nhip, pre)" by (cases r)
  with (update_arg_wf r) have wf1: "flag = val"
    and wf2: "(dsn = 0) = (dsk = unk)"
    and wf3: "dsk = unk  $\implies$  (hops = 1)"
  unfolding update_arg_wf_def by auto

```

```

have "P (dsn, dsk, flag, hops, nhip, pre)"
proof (cases dsk)
  assume "dsk = unk"
  moreover with wf2 wf3 have "dsn = 0" and "hops = Suc 0" by auto
  ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c1)
next
  assume "dsk = kno"
  moreover with wf2 have "dsn > 0" by simp
  ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c2)
qed
with * show "P r" by simp
qed

```

definition update :: "rt ⇒ ip ⇒ r ⇒ rt"

```

where
  "update rt ip r ≡
  case σroute(rt, ip) of
    None ⇒ rt (ip ↦ r)
  | Some s ⇒
    if π2(s) < π2(r) then rt (ip ↦ addpre r (π7(s)))
    else if π2(s) = π2(r) ∧ (π5(s) > π5(r) ∨ π4(s) = inv)
      then rt (ip ↦ addpre r (π7(s)))
    else if π3(r) = unk
      then rt (ip ↦ (π2(s), snd (addpre r (π7(s))))
    else rt (ip ↦ addpre s (π7(r)))"

```

lemma update\_simps [simp]:

```

fixes r s nrt nr nr' ns rt ip
defines "s ≡ the σroute(rt, ip)"
  and "nr ≡ addpre r (π7(s))"
  and "nr' ≡ (π2(s), π3(nr), π4(nr), π5(nr), π6(nr), π7(nr))"
  and "ns ≡ addpre s (π7(r))"

```

shows

```

"[[ip ∉ kD(rt)] ⇒ update rt ip r = rt (ip ↦ r)"
"[[ip ∈ kD(rt); sqn rt ip < π2(r)] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); sqn rt ip = π2(r);
  the (dhops rt ip) > π5(r)] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); sqn rt ip = π2(r);
  flag rt ip = Some inv] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); π3(r) = unk; (π2(r) = 0) = (π3(r) = unk)] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); sqn rt ip ≥ π2(r); π3(r) = kno;
  sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val ]
  ⇒ update rt ip r = rt (ip ↦ ns)"

```

proof -

```

assume "ip ∉ kD(rt)"
hence "σroute(rt, ip) = None" ..
thus "update rt ip r = rt (ip ↦ r)"
  unfolding update_def by simp
next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip < π2(r)"
  from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
  with ⟨sqn rt ip < π2(r)⟩ show "update rt ip r = rt (ip ↦ nr)"
  unfolding update_def nr_def s_def by auto
next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip = π2(r)"
  and "the (dhops rt ip) > π5(r)"
  from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
  by (metis kD_Some)
  with ⟨sqn rt ip = π2(r)⟩ and ⟨the (dhops rt ip) > π5(r)⟩

```

```

  show "update rt ip r = rt (ip ↦ nr)"
    unfolding update_def nr_def s_def by auto
next
  assume "ip ∈ kD(rt)"
    and "sqn rt ip = π2(r)"
    and "flag rt ip = Some inv"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with ⟨sqn rt ip = π2(r)⟩ and ⟨flag rt ip = Some inv⟩
  show "update rt ip r = rt (ip ↦ nr)"
    unfolding update_def nr_def s_def by auto
next
  assume "ip ∈ kD(rt)"
    and "π3(r) = unk"
    and "(π2(r) = 0) = (π3(r) = unk)"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with ⟨(π2(r) = 0) = (π3(r) = unk)⟩ and ⟨π3(r) = unk⟩
  show "update rt ip r = rt (ip ↦ nr)"
    unfolding update_def nr'_def nr_def s_def
    by (cases r) simp
next
  assume "ip ∈ kD(rt)"
    and otherassms: "sqn rt ip ≥ π2(r)"
    "π3(r) = kno"
    "sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with otherassms show "update rt ip r = rt (ip ↦ ns)"
    unfolding update_def ns_def s_def by auto
qed

```

lemma update\_cases [elim]:

```

  assumes "(π2(r) = 0) = (π3(r) = unk)"
    and c1: "[ip ∉ kD(rt)] ⇒ P (rt (ip ↦ r))"

    and c2: "[ip ∈ kD(rt); sqn rt ip < π2(r)]
      ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
    and c3: "[ip ∈ kD(rt); sqn rt ip = π2(r); the (dhops rt ip) > π5(r)]
      ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
    and c4: "[ip ∈ kD(rt); sqn rt ip = π2(r); the (flag rt ip) = inv]
      ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
    and c5: "[ip ∈ kD(rt); π3(r) = unk]
      ⇒ P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r),
        π4(r), π5(r), π6(r), π7(addpre r (π7(the σroute(rt, ip))))))"
    and c6: "[ip ∈ kD(rt); sqn rt ip ≥ π2(r); π3(r) = kno;
      sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val]
      ⇒ P (rt (ip ↦ addpre (the σroute(rt, ip)) (π7(r))))"

```

shows "(P (update rt ip r))"

proof (cases "ip ∈ kD(rt)")

assume "ip ∉ kD(rt)"

with c1 show ?thesis

by simp

next

assume "ip ∈ kD(rt)"

moreover then obtain dsn dsk fl hops nhip pre

where rteq: "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"

by (metis kD\_Some)

moreover obtain dsn' dsk' fl' hops' nhip' pre'

where req: "r = (dsn', dsk', fl', hops', nhip', pre)'"

by (cases r) metis

```

ultimately show ?thesis
using ⟨(π2(r) = 0) = (π3(r) = unk)⟩
  c2 [OF ⟨ip ∈ kD(rt)⟩]
  c3 [OF ⟨ip ∈ kD(rt)⟩]
  c4 [OF ⟨ip ∈ kD(rt)⟩]
  c5 [OF ⟨ip ∈ kD(rt)⟩]
  c6 [OF ⟨ip ∈ kD(rt)⟩]
unfolding update_def sqn_def by auto
qed

```

lemma update\_cases\_kD:

```

assumes "(π2(r) = 0) = (π3(r) = unk)"
  and "ip ∈ kD(rt)"
  and c2: "sqn rt ip < π2(r) ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
  and c3: "⟦sqn rt ip = π2(r); the (dhops rt ip) > π5(r)⟧
    ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
  and c4: "⟦sqn rt ip = π2(r); the (flag rt ip) = inv⟧
    ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
  and c5: "π3(r) = unk ⇒ P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r),
    π4(r), π5(r), π6(r),
    π7(addpre r (π7(the σroute(rt, ip))))))"
  and c6: "⟦sqn rt ip ≥ π2(r); π3(r) = kno;
    sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val⟧
    ⇒ P (rt (ip ↦ addpre (the σroute(rt, ip)) (π7(r))))"

```

shows "(P (update rt ip r))"

using assms(1) proof (rule update\_cases)

assume "sqn rt ip < π<sub>2</sub>(r)"

thus "P (rt(ip ↦ addpre r (π<sub>7</sub>(the (rt ip)))))" by (rule c2)

next

assume "sqn rt ip = π<sub>2</sub>(r)"

and "the (dhops rt ip) > π<sub>5</sub>(r)"

thus "P (rt(ip ↦ addpre r (π<sub>7</sub>(the (rt ip)))))"

by (rule c3)

next

assume "sqn rt ip = π<sub>2</sub>(r)"

and "the (flag rt ip) = inv"

thus "P (rt(ip ↦ addpre r (π<sub>7</sub>(the (rt ip)))))"

by (rule c4)

next

assume "π<sub>3</sub>(r) = unk"

thus "P (rt (ip ↦ (π<sub>2</sub>(the σ<sub>route</sub>(rt, ip)), π<sub>3</sub>(r), π<sub>4</sub>(r), π<sub>5</sub>(r), π<sub>6</sub>(r),
 π<sub>7</sub>(addpre r (π<sub>7</sub>(the (rt ip))))))"

by (rule c5)

next

assume "sqn rt ip ≥ π<sub>2</sub>(r)"

and "π<sub>3</sub>(r) = kno"

and "sqn rt ip = π<sub>2</sub>(r) ⇒ the (dhops rt ip) ≤ π<sub>5</sub>(r) ∧ the (flag rt ip) = val"

thus "P (rt (ip ↦ addpre (the (rt ip)) (π<sub>7</sub>(r))))"

by (rule c6)

qed (simp add: ⟨ip ∈ kD(rt)⟩)

lemma in\_kD\_after\_update [simp]:

fixes rt nip dsn dsk flag hops nhip pre

shows "kD (update rt nip (dsn, dsk, flag, hops, nhip, pre)) = insert nip (kD rt)"

unfolding update\_def

by (cases "rt nip") auto

lemma nhop\_of\_update [simp]:

fixes rt dip dsn dsk flag hops nhip

assumes "rt ≠ update rt dip (dsn, dsk, flag, hops, nhip, {})"

shows "the (nhop (update rt dip (dsn, dsk, flag, hops, nhip, {})) dip) = nhip"

proof -

from assms

have update\_neq: "∧v. rt dip = Some v ⇒

```

      update rt dip (dsn, dsk, flag, hops, nhip, {})
        ≠ rt(dip ↦ addpre (the (rt dip)) (π7 (dsn, dsk, flag, hops, nhip, {})))"
    by auto
show ?thesis
proof (cases "rt dip = None")
  assume "rt dip = None"
  thus "?thesis" unfolding update_def by clarsimp
next
  assume "rt dip ≠ None"
  then obtain v where "rt dip = Some v" by (metis not_None_eq)
  with update_neq [OF this] show ?thesis
    unfolding update_def by auto
qed
qed

lemma sqn_if_updated:
  fixes rip v rt ip
  shows "sqn (λx. if x = rip then Some v else rt x) ip
    = (if ip = rip then π2(v) else sqn rt ip)"
  unfolding sqn_def by simp

lemma update_sqn [simp]:
  fixes rt dip rip dsn dsk hops nhip pre
  assumes "(dsn = 0) = (dsk = unk)"
  shows "sqn rt dip ≤ sqn (update rt rip (dsn, dsk, val, hops, nhip, pre)) dip"
  proof (rule update_cases)
    show "(π2 (dsn, dsk, val, hops, nhip, pre) = 0) = (π3 (dsn, dsk, val, hops, nhip, pre) = unk)"
      by simp (rule assms)
  qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
  fixes rt ip ip' dsn dsk flag hops nhip pre
  assumes "1 ≤ hops"
  shows "sqn rt ip ≤ sqn (update rt ip' (dsn, dsk, flag, hops, nhip, pre)) ip"
  using assms unfolding update_def sqn_def
  by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
  fixes rt dsn dsk flag hops ip rip nhip pre
  assumes ex: "∀ip∈kD rt. the (dhops rt ip) ≥ 1"
    and ip: "(ip = rip ∧ Suc 0 ≤ hops) ∨ (ip ≠ rip ∧ ip∈kD rt)"
  shows "Suc 0 ≤ the (dhops (update rt rip (dsn, dsk, flag, hops, nhip, pre)) ip)"
  using ip proof
    assume "ip = rip ∧ Suc 0 ≤ hops" thus ?thesis
      unfolding update_def using ex
      by (cases "rip ∈ kD rt") (drule(1) bspec, auto)
  next
    assume "ip ≠ rip ∧ ip∈kD rt" thus ?thesis
      using ex unfolding update_def
      by (cases "rip∈kD rt") auto
  qed

lemma update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "(update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma nhop_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = nhop rt ip"
  using assms unfolding update_def

```

```

by (clarsimp split: option.split)

lemma dhops_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip  $\neq$  dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = dhops rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
  " $\wedge$ rt ip dsn dsk flag hops nhip pre. sqn (rt(ip  $\mapsto$  v)) ip =  $\pi_2$ (v)"
  unfolding sqn_def by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip
  assumes "rt  $\neq$  update rt dip (osn, kno, val, hops, nhip, {})"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nhip, {})) dip) = hops"
  using assms unfolding update_def
  by (clarsimp split: option.split_asm option.split split_if_asm) auto

lemma nhop_update_unk_val [simp]:
  " $\wedge$ rt dip ip dsn hops npre.
  the (nhop (update rt dip (dsn, unk, val, hops, ip, npre)) dip) = ip"
  unfolding update_def by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
  assumes "update rt dip (dsn, dsk, flg, hops, sip, {})  $\neq$  rt"
  shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
  using assms unfolding update_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma update_rt_split_asm:
  " $\wedge$ rt ip dsn dsk flag hops sip.
  P (update rt ip (dsn, dsk, flag, hops, sip, {}))
  =
  ( $\neg$ (rt = update rt ip (dsn, dsk, flag, hops, sip, {}))  $\wedge$   $\neg$ P rt
   $\vee$  rt  $\neq$  update rt ip (dsn, dsk, flag, hops, sip, {})
   $\wedge$   $\neg$ P (update rt ip (dsn, dsk, flag, hops, sip, {})))"
  by auto

lemma sqn_update [simp]: " $\wedge$ rt dip dsn flg hops sip.
rt  $\neq$  update rt dip (dsn, kno, flg, hops, sip, {})
 $\implies$  sqn (update rt dip (dsn, kno, flg, hops, sip, {})) dip = dsn"
unfolding update_def by (clarsimp split: option.split split_if_asm) auto

lemma sqnf_update [simp]: " $\wedge$ rt dip dsn dsk flg hops sip.
rt  $\neq$  update rt dip (dsn, dsk, flg, hops, sip, {})
 $\implies$  sqnf (update rt dip (dsn, dsk, flg, hops, sip, {})) dip = dsk"
unfolding update_def sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

lemma update_kno_dsn_greater_zero:
  " $\wedge$ rt dip ip dsn hops npre. 1  $\leq$  dsn  $\implies$  1  $\leq$  (sqn (update rt dip (dsn, kno, val, hops, ip, npre)) dip)"
  unfolding update_def
  by (clarsimp split: option.splits)

lemma proj3_update [simp]: " $\wedge$ rt dip dsn dsk flg hops sip.
rt  $\neq$  update rt dip (dsn, dsk, flg, hops, sip, {})
 $\implies$   $\pi_3$ (the (update rt dip (dsn, dsk, flg, hops, sip, {}) dip)) = dsk"
unfolding update_def sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

lemma nhop_update_changed_kno_val [simp]: " $\wedge$ rt ip dsn dsk hops nhip.

```

```

rt ≠ update rt ip (dsn, kno, val, hops, nhip, {})
  ⇒ the (nhop (update rt ip (dsn, kno, val, hops, nhip, {})) ip) = nhip"
using assms unfolding update_def
by (clarsimp split: option.split_asm option.split_split_if_asm) auto

```

```

lemma flag_update [simp]: "∧rt dip dsn flg hops sip.
rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
  ⇒ the (flag (update rt dip (dsn, kno, flg, hops, sip, {})) dip) = flg"
unfolding update_def
by (clarsimp split: option.split_split_if_asm) auto

```

```

lemma the_flag_Some [dest!]:
  fixes ip rt
  assumes "the (flag rt ip) = x"
    and "ip ∈ kD rt"
  shows "flag rt ip = Some x"
using assms by auto

```

```

lemma kD_update_unchanged [dest]:
  fixes rt dip dsn dsk flag hops nhip pre
  assumes "rt = update rt dip (dsn, dsk, flag, hops, nhip, pre)"
  shows "dip ∈ kD(rt)"
proof -
  have "dip ∈ kD(update rt dip (dsn, dsk, flag, hops, nhip, pre))" by simp
  with assms show ?thesis by simp
qed

```

```

lemma nhop_update [simp]: "∧rt dip dsn dsk flg hops sip.
rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
  ⇒ the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
unfolding update_def sqnf_def
by (clarsimp split: option.splits_split_if_asm) auto

```

```

lemma sqn_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqn rt ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.splits) auto

```

```

lemma sqnf_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "sqnf (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqnf rt ip"
using assms unfolding update_def sqnf_def
by (clarsimp split: option.splits) auto

```

```

lemma vD_update_val [dest]:
  "∧dip rt dip' dsn dsk hops nhip pre.
dip ∈ vD(update rt dip' (dsn, dsk, val, hops, nhip, pre)) ⇒ (dip ∈ vD(rt) ∨ dip=dip'"
unfolding update_def vD_def by (clarsimp split: option.split_asm split_if_asm)

```

## Invalidating route entries

```

definition invalidate :: "rt ⇒ (ip → sqn) ⇒ rt"
where "invalidate rt dests ≡
λip. case (rt ip, dests ip) of
  (None, _) ⇒ None
| (Some s, None) ⇒ Some s
| (Some (_, dsk, _, hops, nhip, pre), Some rsn) ⇒
  Some (rsn, dsk, inv, hops, nhip, pre)"

```

```

lemma proj3_invalidate [simp]:
  "∧dip. π3(the ((invalidate rt dests) dip)) = π3(the (rt dip))"

```

unfolding invalidate\_def by (clarsimp split: option.split)

lemma proj5\_invalidate [simp]:

" $\bigwedge dip. \pi_5(\text{the } ((\text{invalidate } rt \text{ dests}) \text{ dip})) = \pi_5(\text{the } (rt \text{ dip}))$ "  
unfolding invalidate\_def by (clarsimp split: option.split)

lemma proj6\_invalidate [simp]:

" $\bigwedge dip. \pi_6(\text{the } ((\text{invalidate } rt \text{ dests}) \text{ dip})) = \pi_6(\text{the } (rt \text{ dip}))$ "  
unfolding invalidate\_def by (clarsimp split: option.split)

lemma proj7\_invalidate [simp]:

" $\bigwedge dip. \pi_7(\text{the } ((\text{invalidate } rt \text{ dests}) \text{ dip})) = \pi_7(\text{the } (rt \text{ dip}))$ "  
unfolding invalidate\_def by (clarsimp split: option.split)

### 1.1.5 Route Requests

lemma invalidate\_kD\_inv [simp]:

" $\bigwedge rt \text{ dests}. kD (\text{invalidate } rt \text{ dests}) = kD \text{ rt}$ "  
unfolding invalidate\_def kD\_def  
by (simp split: option.split)

lemma invalidate\_sqn:

fixes rt dip dests  
assumes " $\forall rsn. \text{dests } dip = \text{Some } rsn \longrightarrow \text{sqn } rt \text{ dip} \leq rsn$ "  
shows " $\text{sqn } rt \text{ dip} \leq \text{sqn } (\text{invalidate } rt \text{ dests}) \text{ dip}$ "  
proof (cases "dip  $\notin$  kD(rt)")  
 assume " $\neg dip \notin kD(rt)$ "  
 hence "dip  $\in$  kD(rt)" by simp  
 then obtain dsn dsk flag hops nhip pre where "rt dip = Some (dsn, dsk, flag, hops, nhip, pre)"  
 by (metis kD\_Some)  
 with assms show " $\text{sqn } rt \text{ dip} \leq \text{sqn } (\text{invalidate } rt \text{ dests}) \text{ dip}$ "  
 by (cases "dests dip") (auto simp add: invalidate\_def sqn\_def)  
qed simp

lemma sqn\_invalidate\_in\_dests [simp]:

fixes dests ipa rsn rt  
assumes "dests ipa = Some rsn"  
 and "ipa  $\in$  kD(rt)"  
 shows " $\text{sqn } (\text{invalidate } rt \text{ dests}) \text{ ipa} = rsn$ "  
unfolding invalidate\_def sqn\_def  
using assms(1) assms(2) [THEN kD\_Some]  
by clarsimp

lemma dhops\_invalidate [simp]:

" $\bigwedge dip. \text{the } (\text{dhops } (\text{invalidate } rt \text{ dests}) \text{ dip}) = \text{the } (\text{dhops } rt \text{ dip})$ "  
unfolding invalidate\_def by (clarsimp split: option.split)

lemma sqnf\_invalidate [simp]:

" $\bigwedge dip. \text{sqnf } (\text{invalidate } (rt \ \xi) \ (\text{dests } \xi)) \text{ dip} = \text{sqnf } (rt \ \xi) \text{ dip}$ "  
unfolding sqnf\_def invalidate\_def by (clarsimp split: option.split)

lemma nhop\_invalidate [simp]:

" $\bigwedge dip. \text{the } (\text{nhop } (\text{invalidate } (rt \ \xi) \ (\text{dests } \xi)) \text{ dip}) = \text{the } (\text{nhop } (rt \ \xi) \text{ dip})$ "  
unfolding invalidate\_def by (clarsimp split: option.split)

lemma invalidate\_other [simp]:

fixes rt dests dip  
assumes "dip  $\notin$  dom(dests)"  
 shows " $\text{invalidate } rt \text{ dests } dip = rt \text{ dip}$ "  
using assms unfolding invalidate\_def  
by (clarsimp split: option.split\_asm)

lemma invalidate\_none [simp]:

fixes rt dests dip



```

assumes "dip $\notin$ kD(rt)"
shows "invalidate rt dests dip = None"
using assms unfolding invalidate_def by clarsimp

lemma vD_invalidate_vD_not_dests:
" $\bigwedge$ dip rt dests. dip $\in$ vD(invalidate rt dests)  $\implies$  dip $\in$ vD(rt)  $\wedge$  dests dip = None"
unfolding invalidate_def vD_def
by (clarsimp split: option.split_asm)

lemma sqn_invalidate_not_in_dests [simp]:
fixes dests dip rt
assumes "dip $\notin$ dom(dests)"
shows "sqn (invalidate rt dests) dip = sqn rt dip"
using assms unfolding sqn_def by simp

lemma invalidate_changes:
fixes rt dests dip dsn dsk flag hops nhip pre
assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip, pre)"
shows " dsn = (case dests dip of None  $\Rightarrow$   $\pi_2$ (the (rt dip)) | Some rsn  $\Rightarrow$  rsn)
 $\wedge$  dsk =  $\pi_3$ (the (rt dip))
 $\wedge$  flag = (if dests dip = None then  $\pi_4$ (the (rt dip)) else inv)
 $\wedge$  hops =  $\pi_5$ (the (rt dip))
 $\wedge$  nhip =  $\pi_6$ (the (rt dip))
 $\wedge$  pre =  $\pi_7$ (the (rt dip))"
using assms unfolding invalidate_def
by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: " $\bigwedge$ dip rt dests. dip $\in$ kD (rt)
 $\implies$   $\pi_3$ (the (invalidate rt dests dip)) =  $\pi_3$ (the (rt dip))"
by (clarsimp simp: invalidate_def kD_def split: option.split)

lemma dests_iD_invalidate [simp]:
assumes "dests ip = Some rsn"
and "ip $\in$ kD(rt)"
shows "ip $\in$ iD(invalidate rt dests)"
using assms(1) assms(2) [THEN kD_Some] unfolding invalidate_def iD_def
by (clarsimp split: option.split)

```

### 1.1.6 Queued Packets

Functions for sending data packets.

```

type_synonym store = "ip  $\rightarrow$  (p  $\times$  data list)"

definition sigma_queue :: "store  $\Rightarrow$  ip  $\Rightarrow$  data list" (" $\sigma_{\text{queue}}'(\_, \_)$ ")
where " $\sigma_{\text{queue}}$ (store, dip)  $\equiv$  case store dip of None  $\Rightarrow$  [] | Some (p, q)  $\Rightarrow$  q"

definition qD :: "store  $\Rightarrow$  ip set"
where "qD  $\equiv$  dom"

definition add :: "data  $\Rightarrow$  ip  $\Rightarrow$  store  $\Rightarrow$  store"
where "add d dip store  $\equiv$  case store dip of
None  $\Rightarrow$  store (dip  $\mapsto$  (req, [d]))
| Some (p, q)  $\Rightarrow$  store (dip  $\mapsto$  (p, q @ [d]))"

lemma qD_add [simp]:
fixes d dip store
shows "qD(add d dip store) = insert dip (qD store)"
unfolding add_def Let_def qD_def
by (clarsimp split: option.split)

definition drop :: "ip  $\Rightarrow$  store  $\rightarrow$  store"
where "drop dip store  $\equiv$ 
map_option ( $\lambda$ (p, q). if tl q = [] then store (dip := None)

```

```
else store (dip ↦ (p, tl q)) (store dip)"
```

```
definition sigma_p_flag :: "store ⇒ ip → p" ("σp-flag'(, _)")
  where "σp-flag(store, dip) ≡ map_option fst (store dip)"
```

```
definition unsetRRF :: "store ⇒ ip ⇒ store"
  where "unsetRRF store dip ≡ case store dip of
    None ⇒ store
    | Some (p, q) ⇒ store (dip ↦ (noreq, q))"
```

```
definition setRRF :: "store ⇒ (ip → sqn) ⇒ store"
  where "setRRF store dests ≡ λdip. if dests dip = None then store dip
    else map_option (λ(, q). (req, q)) (store dip)"
```

### 1.1.7 Comparison with the original technical report

The major differences with the AODV technical report of Fehnker et al are:

1. *nhop* is partial, thus a ‘*the*’ is needed, similarly for *dhops* and *addpreRT*.
2. *precs* is partial.
3.  $\sigma_{p\text{-flag}}(\text{store}, \text{dip})$  is partial.
4. The routing table (*rt*) is modelled as a map ( $\text{ip} \Rightarrow r \text{ option}$ ) rather than a set of 7-tuples, likewise, the *r* is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address (*dip*) is taken from the argument to the function, rather than a part of the result. Well-definedness then follows from the structure of the type and more related facts are available automatically, rather than having to be acquired through tedious proofs.
5. Similar remarks hold for the *dests* mapping passed to *invalidate*, and *store*.

end

## 1.2 AODV protocol messages

```
theory A_Aodv_Message
imports A_Norreqid
begin
```

```
datatype msg =
  Rreq nat ip sqn k ip sqn ip
  | Rrep nat ip sqn ip ip
  | Rerr "ip → sqn" ip
  | Newpkt data ip
  | Pkt data ip ip
```

```
instantiation msg :: msg
```

```
begin
```

```
  definition newpkt_def [simp]: "newpkt ≡ λ(d, dip). Newpkt d dip"
```

```
  definition eq_newpkt_def: "eq_newpkt m ≡ case m of Newpkt d dip ⇒ True | _ ⇒ False"
```

```
  instance by intro_classes (simp add: eq_newpkt_def)
```

```
end
```

The *msg* type models the different messages used within AODV. The instantiation as a *msg* is a technicality due to the special treatment of *newpkt* messages in the AWN SOS rules. This use of classes allows a clean separation of the AWN-specific definitions and these AODV-specific definitions.

```
definition rreq :: "nat × ip × sqn × k × ip × sqn × ip ⇒ msg"
  where "rreq ≡ λ(hops, dip, dsn, dsk, oip, osn, sip).
    Rreq hops dip dsn dsk oip osn sip"
```

```
lemma rreq_simp [simp]:
```

```
"rreq(hops, dip, dsn, dsk, oip, osn, sip) = Rreq hops dip dsn dsk oip osn sip"
```

```

unfolding rreq_def by simp

definition rrep :: "nat × ip × sqn × ip × ip ⇒ msg"
  where "rrep ≡ λ(hops, dip, dsn, oip, sip). Rrep hops dip dsn oip sip"

lemma rrep_simp [simp]:
  "rrep(hops, dip, dsn, oip, sip) = Rrep hops dip dsn oip sip"
  unfolding rrep_def by simp

definition rerr :: "(ip → sqn) × ip ⇒ msg"
  where "rerr ≡ λ(dests, sip). Rerr dests sip"

lemma rerr_simp [simp]:
  "rerr(dests, sip) = Rerr dests sip"
  unfolding rerr_def by simp

lemma not_eq_newpkt_rreq [simp]: "¬eq_newpkt (Rreq hops dip dsn dsk oip osn sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_rrep [simp]: "¬eq_newpkt (Rrep hops dip dsn oip sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_rerr [simp]: "¬eq_newpkt (Rerr dests sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_pkt [simp]: "¬eq_newpkt (Pkt d dip sip)"
  unfolding eq_newpkt_def by simp

definition pkt :: "data × ip × ip ⇒ msg"
  where "pkt ≡ λ(d, dip, sip). Pkt d dip sip"

lemma pkt_simp [simp]:
  "pkt(d, dip, sip) = Pkt d dip sip"
  unfolding pkt_def by simp

end

```

## 1.3 The AODV protocol

```

theory A_Aodv
imports A_Aodv_Data A_Aodv_Message
  "../.../AWN/AWN_SOS_Labels" "../.../AWN/AWN_Invariants"
begin

```

### 1.3.1 Data state

```

record state =
  ip      :: "ip"
  sn      :: "sqn"
  rt      :: "rt"
  rreqs   :: "(ip × sqn) set"
  store   :: "store"

  msg     :: "msg"
  data    :: "data"
  dests   :: "ip → sqn"
  pre     :: "ip set"
  dip     :: "ip"
  oip     :: "ip"
  hops    :: "nat"
  dsn     :: "sqn"
  dsk     :: "k"
  osn     :: "sqn"
  sip     :: "ip"

```

abbreviation aadv\_init :: "ip  $\Rightarrow$  state"

```
where "aadv_init i  $\equiv$  (
  ip = i,
  sn = 1,
  rt = empty,
  rreqs = {},
  store = empty,

  msg    = (SOME x. True),
  data   = (SOME x. True),
  dests  = (SOME x. True),
  pre    = (SOME x. True),
  dip    = (SOME x. True),
  oip    = (SOME x. True),
  hops   = (SOME x. True),
  dsn    = (SOME x. True),
  dsk    = (SOME x. True),
  osn    = (SOME x. True),
  sip    = (SOME x. x  $\neq$  i)
)"
```

lemma some\_neq\_not\_eq [simp]: " $\neg((\text{SOME } x :: \text{nat. } x \neq i) = i)$ "  
by (subst some\_eq\_ex) (metis zero\_neq\_numeral)

definition clear\_locals :: "state  $\Rightarrow$  state"

```
where "clear_locals  $\xi$  =  $\xi$  (
  msg    := (SOME x. True),
  data   := (SOME x. True),
  dests  := (SOME x. True),
  pre    := (SOME x. True),
  dip    := (SOME x. True),
  oip    := (SOME x. True),
  hops   := (SOME x. True),
  dsn    := (SOME x. True),
  dsk    := (SOME x. True),
  osn    := (SOME x. True),
  sip    := (SOME x. x  $\neq$  ip  $\xi$ )
)"
```

lemma clear\_locals\_sip\_not\_ip [simp]: " $\neg(\text{sip } (\text{clear\_locals } \xi) = \text{ip } \xi)$ "  
unfolding clear\_locals\_def by simp

lemma clear\_locals\_but\_not\_globals [simp]:

```
"ip (clear_locals  $\xi$ ) = ip  $\xi$ "
"sn (clear_locals  $\xi$ ) = sn  $\xi$ "
"rt (clear_locals  $\xi$ ) = rt  $\xi$ "
"rreqs (clear_locals  $\xi$ ) = rreqs  $\xi$ "
"store (clear_locals  $\xi$ ) = store  $\xi$ "
unfolding clear_locals_def by auto
```

### 1.3.2 Auxilliary message handling definitions

definition is\_newpkt

```
where "is_newpkt  $\xi \equiv$  case msg  $\xi$  of
  Newpkt data' dip'  $\Rightarrow$  {  $\xi$ (data := data', dip := dip') }
  | _  $\Rightarrow$  {}"
```

definition is\_pkt

```
where "is_pkt  $\xi \equiv$  case msg  $\xi$  of
  Pkt data' dip' oip'  $\Rightarrow$  {  $\xi$ (data := data', dip := dip', oip := oip' ) }
  | _  $\Rightarrow$  {}"
```

definition is\_rreq

```

where "is_rreq  $\xi \equiv$  case msg  $\xi$  of
      Rreq hops' dip' dsn' dsk' oip' osn' sip'  $\Rightarrow$ 
        {  $\xi$ (| hops := hops', dip := dip', dsn := dsn',
            dsk := dsk', oip := oip', osn := osn', sip := sip' |) }
      | _  $\Rightarrow$  {}"

```

```

lemma is_rreq_asm [dest!]:
  assumes " $\xi' \in$  is_rreq  $\xi$ "
  shows "( $\exists$  hops' rreqid' dip' dsn' dsk' oip' osn' sip'.
        msg  $\xi =$  Rreq hops' dip' dsn' dsk' oip' osn' sip'  $\wedge$ 
         $\xi' = \xi$ (| hops := hops', dip := dip', dsn := dsn',
            dsk := dsk', oip := oip', osn := osn', sip := sip' |))"
  using assms unfolding is_rreq_def
  by (cases "msg  $\xi$ ") simp_all

```

```

definition is_rrep
where "is_rrep  $\xi \equiv$  case msg  $\xi$  of
      Rrep hops' dip' dsn' oip' sip'  $\Rightarrow$ 
        {  $\xi$ (| hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' |) }
      | _  $\Rightarrow$  {}"

```

```

lemma is_rrep_asm [dest!]:
  assumes " $\xi' \in$  is_rrep  $\xi$ "
  shows "( $\exists$  hops' dip' dsn' oip' sip'.
        msg  $\xi =$  Rrep hops' dip' dsn' oip' sip'  $\wedge$ 
         $\xi' = \xi$ (| hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' |))"
  using assms unfolding is_rrep_def
  by (cases "msg  $\xi$ ") simp_all

```

```

definition is_rerr
where "is_rerr  $\xi \equiv$  case msg  $\xi$  of
      Rerr dests' sip'  $\Rightarrow$  {  $\xi$ (| dests := dests', sip := sip' |) }
      | _  $\Rightarrow$  {}"

```

```

lemma is_rerr_asm [dest!]:
  assumes " $\xi' \in$  is_rerr  $\xi$ "
  shows "( $\exists$  dests' sip'.
        msg  $\xi =$  Rerr dests' sip'  $\wedge$ 
         $\xi' = \xi$ (| dests := dests', sip := sip' |))"
  using assms unfolding is_rerr_def
  by (cases "msg  $\xi$ ") simp_all

```

```

lemmas is_msg_defs =
  is_rerr_def is_rrep_def is_rreq_def is_pkt_def is_newpkt_def

```

```

lemma is_msg_inv_ip [simp]:
  " $\xi' \in$  is_rerr  $\xi \implies$  ip  $\xi' =$  ip  $\xi$ "
  " $\xi' \in$  is_rrep  $\xi \implies$  ip  $\xi' =$  ip  $\xi$ "
  " $\xi' \in$  is_rreq  $\xi \implies$  ip  $\xi' =$  ip  $\xi$ "
  " $\xi' \in$  is_pkt  $\xi \implies$  ip  $\xi' =$  ip  $\xi$ "
  " $\xi' \in$  is_newpkt  $\xi \implies$  ip  $\xi' =$  ip  $\xi$ "
  unfolding is_msg_defs
  by (cases "msg  $\xi$ ", clarsimp)+

```

```

lemma is_msg_inv_sn [simp]:
  " $\xi' \in$  is_rerr  $\xi \implies$  sn  $\xi' =$  sn  $\xi$ "
  " $\xi' \in$  is_rrep  $\xi \implies$  sn  $\xi' =$  sn  $\xi$ "
  " $\xi' \in$  is_rreq  $\xi \implies$  sn  $\xi' =$  sn  $\xi$ "
  " $\xi' \in$  is_pkt  $\xi \implies$  sn  $\xi' =$  sn  $\xi$ "
  " $\xi' \in$  is_newpkt  $\xi \implies$  sn  $\xi' =$  sn  $\xi$ "
  unfolding is_msg_defs
  by (cases "msg  $\xi$ ", clarsimp)+

```

```

lemma is_msg_inv_rt [simp]:

```

```

" $\xi'$   $\in$  is_rerr  $\xi$   $\implies$  rt  $\xi'$  = rt  $\xi$ "
" $\xi'$   $\in$  is_rrep  $\xi$   $\implies$  rt  $\xi'$  = rt  $\xi$ "
" $\xi'$   $\in$  is_rreq  $\xi$   $\implies$  rt  $\xi'$  = rt  $\xi$ "
" $\xi'$   $\in$  is_pkt  $\xi$   $\implies$  rt  $\xi'$  = rt  $\xi$ "
" $\xi'$   $\in$  is_newpkt  $\xi$   $\implies$  rt  $\xi'$  = rt  $\xi$ "
unfolding is_msg_defs
by (cases "msg  $\xi$ ", clarsimp+)+

```

```

lemma is_msg_inv_rreqs [simp]:
" $\xi'$   $\in$  is_rerr  $\xi$   $\implies$  rreqs  $\xi'$  = rreqs  $\xi$ "
" $\xi'$   $\in$  is_rrep  $\xi$   $\implies$  rreqs  $\xi'$  = rreqs  $\xi$ "
" $\xi'$   $\in$  is_rreq  $\xi$   $\implies$  rreqs  $\xi'$  = rreqs  $\xi$ "
" $\xi'$   $\in$  is_pkt  $\xi$   $\implies$  rreqs  $\xi'$  = rreqs  $\xi$ "
" $\xi'$   $\in$  is_newpkt  $\xi$   $\implies$  rreqs  $\xi'$  = rreqs  $\xi$ "
unfolding is_msg_defs
by (cases "msg  $\xi$ ", clarsimp+)+

```

```

lemma is_msg_inv_store [simp]:
" $\xi'$   $\in$  is_rerr  $\xi$   $\implies$  store  $\xi'$  = store  $\xi$ "
" $\xi'$   $\in$  is_rrep  $\xi$   $\implies$  store  $\xi'$  = store  $\xi$ "
" $\xi'$   $\in$  is_rreq  $\xi$   $\implies$  store  $\xi'$  = store  $\xi$ "
" $\xi'$   $\in$  is_pkt  $\xi$   $\implies$  store  $\xi'$  = store  $\xi$ "
" $\xi'$   $\in$  is_newpkt  $\xi$   $\implies$  store  $\xi'$  = store  $\xi$ "
unfolding is_msg_defs
by (cases "msg  $\xi$ ", clarsimp+)+

```

```

lemma is_msg_inv_sip [simp]:
" $\xi'$   $\in$  is_pkt  $\xi$   $\implies$  sip  $\xi'$  = sip  $\xi$ "
" $\xi'$   $\in$  is_newpkt  $\xi$   $\implies$  sip  $\xi'$  = sip  $\xi$ "
unfolding is_msg_defs
by (cases "msg  $\xi$ ", clarsimp+)+

```

### 1.3.3 The protocol process

```

datatype pseqp =
  PAadv
  | PNewPkt
  | PPkt
  | PRreq
  | PRrep
  | PRerr

```

```

fun nat_of_seqp :: "pseqp  $\Rightarrow$  nat"
where
  "nat_of_seqp PAadv = 1"
  | "nat_of_seqp PPkt = 2"
  | "nat_of_seqp PNewPkt = 3"
  | "nat_of_seqp PRreq = 4"
  | "nat_of_seqp PRrep = 5"
  | "nat_of_seqp PRerr = 6"

```

```

instantiation "pseqp" :: ord
begin
definition less_eq_seqp [iff]: "l1  $\leq$  l2 = (nat_of_seqp l1  $\leq$  nat_of_seqp l2)"
definition less_seqp [iff]: "l1 < l2 = (nat_of_seqp l1 < nat_of_seqp l2)"
instance ..
end

```

```

abbreviation AODV
where
  "AODV  $\equiv$   $\lambda$ _. [[clear_locals]] call(PAadv)"

```

```

abbreviation PKT
where

```

```

"PKT args ≡

[[ξ. let (data, dip, oip) = args ξ in
  (clear_locals ξ) (| data := data, dip := dip, oip := oip |)]
call(PPkt)"

abbreviation NEWPKT
where
  "NEWPKT args ≡
  [[ξ. let (data, dip) = args ξ in
    (clear_locals ξ) (| data := data, dip := dip |)]
  call(PNewPkt)"

abbreviation RREQ
where
  "RREQ args ≡
  [[ξ. let (hops, dip, dsn, dsk, oip, osn, sip) = args ξ in
    (clear_locals ξ) (| hops := hops, dip := dip,
      dsn := dsn, dsk := dsk, oip := oip,
      osn := osn, sip := sip |)]
  call(PRreq)"

abbreviation RREP
where
  "RREP args ≡
  [[ξ. let (hops, dip, dsn, oip, sip) = args ξ in
    (clear_locals ξ) (| hops := hops, dip := dip, dsn := dsn,
      oip := oip, sip := sip |)]
  call(PRrep)"

abbreviation RERR
where
  "RERR args ≡
  [[ξ. let (dests, sip) = args ξ in
    (clear_locals ξ) (| dests := dests, sip := sip |)]
  call(PRerr)"

fun ΓAODV :: "(state, msg, pseqp, pseqp label) seqp_env"
where
  "ΓAODV PAodv = labelled PAodv (
  receive(λmsg' ξ. ξ (| msg := msg' |)).
  (
  <is_newpkt> NEWPKT(λξ. (data ξ, ip ξ))
  ⊕ <is_pkt> PKT(λξ. (data ξ, dip ξ, oip ξ))
  ⊕ <is_rreq>
  [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, {f}) |)]
  RREQ(λξ. (hops ξ, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, sip ξ))
  ⊕ <is_rrep>
  [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, {f}) |)]
  RREP(λξ. (hops ξ, dip ξ, dsn ξ, oip ξ, sip ξ))
  ⊕ <is_rerr>
  [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, {f}) |)]
  RERR(λξ. (dests ξ, sip ξ))
  )
  ⊕ <λξ. { ξ(| dip := dip |) | dip. dip ∈ qD(store ξ) ∩ vD(rt ξ) }>
  [[ξ. ξ (| data := hd(σqueue(store ξ, dip ξ)) |)]
  unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, ip ξ)).
  [[ξ. ξ (| store := the (drop (dip ξ) (store ξ)) |)]
  AODV()
  ▷ [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
    then Some (inc (sqn (rt ξ) rip)) else None) |)]
  [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]
  [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]
  [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } |)]
  [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ {f})
    then (dests ξ) rip else None) |)]

```

```

    groupcast( $\lambda\xi$ . pre  $\xi$ ,  $\lambda\xi$ . rerr(dests  $\xi$ , ip  $\xi$ )). AODV()
 $\oplus$   $\langle \lambda\xi$ . {  $\xi$  | dip := dip }
    | dip. dip  $\in$   $qD$ (store  $\xi$ ) -  $vD$ (rt  $\xi$ )  $\wedge$  the ( $\sigma_p$ -flag(store  $\xi$ , dip)) = req }
    [[ $\xi$ .  $\xi$  | store := unsetRRF (store  $\xi$ ) (dip  $\xi$ ) ]]]
    [[ $\xi$ .  $\xi$  | sn := inc (sn  $\xi$ ) ]]]
    [[ $\xi$ .  $\xi$  | rreqs := rreqs  $\xi$   $\cup$  {(ip  $\xi$ , sn  $\xi$ )} ]]]
    broadcast( $\lambda\xi$ . rreq(0, dip  $\xi$ , sqn (rt  $\xi$ ) (dip  $\xi$ ), sqnf (rt  $\xi$ ) (dip  $\xi$ ),
        ip  $\xi$ , sn  $\xi$ , ip  $\xi$ )). AODV()"

| " $\Gamma_{AODV}$  PNewPkt = labelled PNewPkt (
   $\langle \xi$ . dip  $\xi$  = ip  $\xi$  )
  deliver( $\lambda\xi$ . data  $\xi$ ).AODV()
 $\oplus$   $\langle \xi$ . dip  $\xi \neq$  ip  $\xi$  )
  [[ $\xi$ .  $\xi$  | store := add (data  $\xi$ ) (dip  $\xi$ ) (store  $\xi$ ) ]]]
  AODV()"

| " $\Gamma_{AODV}$  PPkt = labelled PPkt (
   $\langle \xi$ . dip  $\xi$  = ip  $\xi$  )
  deliver( $\lambda\xi$ . data  $\xi$ ).AODV()
 $\oplus$   $\langle \xi$ . dip  $\xi \neq$  ip  $\xi$  )
  (
     $\langle \xi$ . dip  $\xi \in vD$  (rt  $\xi$ ) )
    unicast( $\lambda\xi$ . the (nhop (rt  $\xi$ ) (dip  $\xi$ )),  $\lambda\xi$ . pkt(data  $\xi$ , dip  $\xi$ , oip  $\xi$ )).AODV()
     $\triangleright$ 
    [[ $\xi$ .  $\xi$  | dests := ( $\lambda rip$ . if (rip  $\in vD$  (rt  $\xi$ )  $\wedge$  nhop (rt  $\xi$ ) rip = nhop (rt  $\xi$ ) (dip  $\xi$ ))
        then Some (inc (sqn (rt  $\xi$ ) rip)) else None) ]]]
    [[ $\xi$ .  $\xi$  | rt := invalidate (rt  $\xi$ ) (dests  $\xi$ ) ]]]
    [[ $\xi$ .  $\xi$  | store := setRRF (store  $\xi$ ) (dests  $\xi$ ) ]]]
    [[ $\xi$ .  $\xi$  | pre :=  $\bigcup$  { the (precs (rt  $\xi$ ) rip) | rip. rip  $\in$  dom (dests  $\xi$ ) } ]]]
    [[ $\xi$ .  $\xi$  | dests := ( $\lambda rip$ . if ((dests  $\xi$ ) rip  $\neq$  None  $\wedge$  the (precs (rt  $\xi$ ) rip)  $\neq$  {})
        then (dests  $\xi$ ) rip else None) ]]]
    groupcast( $\lambda\xi$ . pre  $\xi$ ,  $\lambda\xi$ . rerr(dests  $\xi$ , ip  $\xi$ )).AODV()
 $\oplus$   $\langle \xi$ . dip  $\xi \notin vD$  (rt  $\xi$ ) )
  (
     $\langle \xi$ . dip  $\xi \in iD$  (rt  $\xi$ ) )
    groupcast( $\lambda\xi$ . the (precs (rt  $\xi$ ) (dip  $\xi$ )),
         $\lambda\xi$ . rerr([dip  $\xi \mapsto$  sqn (rt  $\xi$ ) (dip  $\xi$ )], ip  $\xi$ )). AODV()
 $\oplus$   $\langle \xi$ . dip  $\xi \notin iD$  (rt  $\xi$ ) )
    AODV()
  )
  )"

| " $\Gamma_{AODV}$  PRreq = labelled PRreq (
   $\langle \xi$ . (oip  $\xi$ , osn  $\xi$ )  $\in$  rreqs  $\xi$  )
  AODV()
 $\oplus$   $\langle \xi$ . (oip  $\xi$ , osn  $\xi$ )  $\notin$  rreqs  $\xi$  )
  [[ $\xi$ .  $\xi$  | rt := update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, hops  $\xi$  + 1, sip  $\xi$ , {}) ]]]
  [[ $\xi$ .  $\xi$  | rreqs := rreqs  $\xi$   $\cup$  {(oip  $\xi$ , osn  $\xi$ )} ]]]
  (
     $\langle \xi$ . dip  $\xi$  = ip  $\xi$  )
    [[ $\xi$ .  $\xi$  | sn := max (sn  $\xi$ ) (dsn  $\xi$ ) ]]]
    unicast( $\lambda\xi$ . the (nhop (rt  $\xi$ ) (oip  $\xi$ )),  $\lambda\xi$ . rrep(0, dip  $\xi$ , sn  $\xi$ , oip  $\xi$ , ip  $\xi$ )).AODV()
     $\triangleright$ 
    [[ $\xi$ .  $\xi$  | dests := ( $\lambda rip$ . if (rip  $\in vD$  (rt  $\xi$ )  $\wedge$  nhop (rt  $\xi$ ) rip = nhop (rt  $\xi$ ) (oip  $\xi$ ))
        then Some (inc (sqn (rt  $\xi$ ) rip)) else None) ]]]
    [[ $\xi$ .  $\xi$  | rt := invalidate (rt  $\xi$ ) (dests  $\xi$ ) ]]]
    [[ $\xi$ .  $\xi$  | store := setRRF (store  $\xi$ ) (dests  $\xi$ ) ]]]
    [[ $\xi$ .  $\xi$  | pre :=  $\bigcup$  { the (precs (rt  $\xi$ ) rip) | rip. rip  $\in$  dom (dests  $\xi$ ) } ]]]
    [[ $\xi$ .  $\xi$  | dests := ( $\lambda rip$ . if ((dests  $\xi$ ) rip  $\neq$  None  $\wedge$  the (precs (rt  $\xi$ ) rip)  $\neq$  {})
        then (dests  $\xi$ ) rip else None) ]]]
    groupcast( $\lambda\xi$ . pre  $\xi$ ,  $\lambda\xi$ . rerr(dests  $\xi$ , ip  $\xi$ )).AODV()
 $\oplus$   $\langle \xi$ . dip  $\xi \neq$  ip  $\xi$  )
  (
     $\langle \xi$ . dip  $\xi \in vD$  (rt  $\xi$ )  $\wedge$  dsn  $\xi \leq$  sqn (rt  $\xi$ ) (dip  $\xi$ )  $\wedge$  sqnf (rt  $\xi$ ) (dip  $\xi$ ) = kno )
  )

```



```

[[ξ. ξ (| rt := the (addpreRT (rt ξ) (dip ξ) {sip ξ}) )]]
[[ξ. ξ (| rt := the (addpreRT (rt ξ) (oip ξ) {the (nhop (rt ξ) (dip ξ))}) )]]
unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(the (dhops (rt ξ) (dip ξ)), dip ξ,
      sqn (rt ξ) (dip ξ), oip ξ, ip ξ)).

```

```
AODV()
```

```
▷
```

```
[[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
      then Some (inc (sqn (rt ξ) rip)) else None) )]]
```

```
[[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
```

```
[[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
```

```
[[ξ. ξ (| pre := ⋃{ the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
```

```
[[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ {})
      then (dests ξ) rip else None) )]]
```

```
groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
```

```
⊕ ⟨ξ. dip ξ ∉ vD (rt ξ) ∨ sqn (rt ξ) (dip ξ) < dsn ξ ∨ sqnf (rt ξ) (dip ξ) = unk⟩
```

```
broadcast(λξ. rreq(hops ξ + 1, dip ξ, max (sqn (rt ξ) (dip ξ)) (dsn ξ),
```

```
      dsk ξ, oip ξ, osn ξ, ip ξ)).
```

```
AODV()
```

```
)
```

```
)"
```

```
| "ΓAODV PRrep = labelled PRrep (
```

```
  ⟨ξ. rt ξ ≠ update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {})⟩
```

```
(
```

```
  [[ξ. ξ (| rt := update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {}) )]]
```

```
(
```

```
  ⟨ξ. oip ξ = ip ξ⟩
```

```
AODV()
```

```
⊕ ⟨ξ. oip ξ ≠ ip ξ⟩
```

```
(
```

```
  ⟨ξ. oip ξ ∈ vD (rt ξ)⟩
```

```
[[ξ. ξ (| rt := the (addpreRT (rt ξ) (dip ξ) {the (nhop (rt ξ) (oip ξ))}) )]]
```

```
[[ξ. ξ (| rt := the (addpreRT (rt ξ) (the (nhop (rt ξ) (dip ξ)))
```

```
      {the (nhop (rt ξ) (oip ξ))}) )]]
```

```
unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(hops ξ + 1, dip ξ, dsn ξ, oip ξ, ip ξ)).
```

```
AODV()
```

```
▷
```

```
[[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
      then Some (inc (sqn (rt ξ) rip)) else None) )]]
```

```
[[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
```

```
[[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
```

```
[[ξ. ξ (| pre := ⋃{ the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
```

```
[[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ {})
      then (dests ξ) rip else None) )]]
```

```
groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
```

```
⊕ ⟨ξ. oip ξ ∉ vD (rt ξ)⟩
```

```
AODV()
```

```
)
```

```
)
```

```
)
```

```
⊕ ⟨ξ. rt ξ = update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {})⟩
```

```
AODV()
```

```
)"
```

```
| "ΓAODV PRerr = labelled PRerr (
```

```
  [[ξ. ξ (| dests := (λrip. case (dests ξ) rip of None ⇒ None
```

```
    | Some rsn ⇒ if rip ∈ vD (rt ξ) ∧ the (nhop (rt ξ) rip) = sip ξ
```

```
      ∧ sqn (rt ξ) rip < rsn then Some rsn else None) )]]
```

```
[[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
```

```
[[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
```

```
[[ξ. ξ (| pre := ⋃{ the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
```

```
[[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ {})
      then (dests ξ) rip else None) )]]
```

```
groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)). AODV())"
```

```

declare  $\Gamma_{AODV}.simps$  [simp del, code del]
lemmas  $\Gamma_{AODV}.simps$  [simp, code] =  $\Gamma_{AODV}.simps$  [simplified]

fun  $\Gamma_{AODV}.skeleton$ 
where
  " $\Gamma_{AODV}.skeleton$  PAadv = seqp_skeleton ( $\Gamma_{AODV}$  PAadv)"
  | " $\Gamma_{AODV}.skeleton$  PNewPkt = seqp_skeleton ( $\Gamma_{AODV}$  PNewPkt)"
  | " $\Gamma_{AODV}.skeleton$  PPkt = seqp_skeleton ( $\Gamma_{AODV}$  PPkt)"
  | " $\Gamma_{AODV}.skeleton$  PRreq = seqp_skeleton ( $\Gamma_{AODV}$  PRreq)"
  | " $\Gamma_{AODV}.skeleton$  PRrep = seqp_skeleton ( $\Gamma_{AODV}$  PRrep)"
  | " $\Gamma_{AODV}.skeleton$  PRerr = seqp_skeleton ( $\Gamma_{AODV}$  PRerr)"

lemma  $\Gamma_{AODV}.skeleton\_wf$  [simp]:
  "wellformed  $\Gamma_{AODV}.skeleton$ "
proof (rule, intro allI)
  fix pn pn'
  show "call(pn')  $\notin$  stermsl ( $\Gamma_{AODV}.skeleton$  pn)"
    by (cases pn) simp_all
qed

declare  $\Gamma_{AODV}.skeleton.simps$  [simp del, code del]
lemmas  $\Gamma_{AODV}.skeleton.simps$  [simp, code]
  =  $\Gamma_{AODV}.skeleton.simps$  [simplified  $\Gamma_{AODV}.simps$  seqp_skeleton.simps]

lemma aadv_proc_cases [dest]:
  fixes p pn
  shows "p  $\in$  ctermsl ( $\Gamma_{AODV}$  pn)  $\implies$ 
    (p  $\in$  ctermsl ( $\Gamma_{AODV}$  PAadv)  $\vee$ 
     p  $\in$  ctermsl ( $\Gamma_{AODV}$  PNewPkt)  $\vee$ 
     p  $\in$  ctermsl ( $\Gamma_{AODV}$  PPkt)  $\vee$ 
     p  $\in$  ctermsl ( $\Gamma_{AODV}$  PRreq)  $\vee$ 
     p  $\in$  ctermsl ( $\Gamma_{AODV}$  PRrep)  $\vee$ 
     p  $\in$  ctermsl ( $\Gamma_{AODV}$  PRerr))"

  using assms
  by (cases pn) simp_all

definition  $\sigma_{AODV} :: "ip \implies (state \times (state, msg, pseq, pseq label) seq) set"$ 
where " $\sigma_{AODV} i \equiv \{(aadv\_init\ i, \Gamma_{AODV}\ PAadv)\}$ "

abbreviation paadv
  :: "ip  $\implies (state \times (state, msg, pseq, pseq label) seq, msg seq\_action) automaton"$ 
where
  " $paadv\ i \equiv (\{ init = \sigma_{AODV}\ i, trans = seqp\_sos\ \Gamma_{AODV}\ })"$ "

lemma aadv_trans: "trans (paadv i) = seqp_sos  $\Gamma_{AODV}$ "
  by simp

lemma aadv_control_within [simp]: "control_within  $\Gamma_{AODV}$  (init (paadv i))"
  unfolding  $\sigma_{AODV}.def$  by (rule control_withinI) (auto simp del:  $\Gamma_{AODV}.simps$ )

lemma aadv_wf [simp]:
  "wellformed  $\Gamma_{AODV}$ "
proof (rule, intro allI)
  fix pn pn'
  show "call(pn')  $\notin$  stermsl ( $\Gamma_{AODV}$  pn)"
    by (cases pn) simp_all
qed

lemmas aadv_labels_not_empty [simp] = labels_not_empty [OF aadv_wf]

lemma aadv_ex_label [intro]: " $\exists l. l \in labels\ \Gamma_{AODV}\ p$ "
  by (metis aadv_labels_not_empty all_not_in_conv)

```

```

lemma aadv_ex_labelE [elim]:
  assumes "∀l∈labels ΓAODV p. P l p"
  and "∃p l. P l p ⇒ Q"
  shows "Q"
  using assms by (metis aadv_ex_label)

lemma aadv_simple_labels [simp]: "simple_labels ΓAODV"
proof
  fix pn p
  assume "p∈subterms(ΓAODV pn)"
  thus "∃!l. labels ΓAODV p = {l}"
  by (cases pn) (simp_all cong: seqp_congs | elim disjE)+
qed

lemma σAODV_labels [simp]: "(ξ, p) ∈ σAODV i ⇒ labels ΓAODV p = {PAadv-:0}"
unfolding σAODV_def by simp

lemma aadv_init_kD_empty [simp]:
  "(ξ, p) ∈ σAODV i ⇒ kD (rt ξ) = {}"
unfolding σAODV_def kD_def by simp

lemma aadv_init_sip_not_ip [simp]: "¬(sip (aadv_init i) = i)" by simp

lemma aadv_init_sip_not_ip' [simp]:
  assumes "(ξ, p) ∈ σAODV i"
  shows "sip ξ ≠ ip ξ"
  using assms unfolding σAODV_def by simp

lemma aadv_init_sip_not_i [simp]:
  assumes "(ξ, p) ∈ σAODV i"
  shows "sip ξ ≠ i"
  using assms unfolding σAODV_def by simp

lemma clear_locals_sip_not_ip':
  assumes "ip ξ = i"
  shows "¬(sip (clear_locals ξ) = i)"
  using assms by auto

Stop the simplifier from descending into process terms.
declare seqp_congs [cong]

Configure the main invariant tactic for AODV.
declare
  ΓAODV_simps [cterms_env]
  aadv_proc_cases [ctermsl_cases]
  seq_invariant_ctermsI [OF aadv_wf aadv_control_within aadv_simple_labels aadv_trans,
    cterms_intros]
  seq_step_invariant_ctermsI [OF aadv_wf aadv_control_within aadv_simple_labels aadv_trans,
    cterms_intros]

end

```

## 1.4 Invariant assumptions and properties

```

theory A_Aadv_Predicates
imports A_Aadv
begin

```

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

```

abbreviation not_Pkt :: "msg ⇒ bool"
where "not_Pkt m ≡ case m of Pkt _ _ _ ⇒ False | _ ⇒ True"

definition msg_sender :: "msg ⇒ ip"

```

where "msg\_sender m  $\equiv$  case m of Rreq \_ \_ \_ \_ \_ ipc  $\Rightarrow$  ipc  
 | Rrep \_ \_ \_ \_ ipc  $\Rightarrow$  ipc  
 | Rerr \_ ipc  $\Rightarrow$  ipc  
 | Pkt \_ \_ ipc  $\Rightarrow$  ipc"

lemma msg\_sender\_simps [simp]:

" $\wedge$ hops dip dsn dsk oip osn sip.  
 msg\_sender (Rreq hops dip dsn dsk oip osn sip) = sip"  
 " $\wedge$ hops dip dsn oip sip. msg\_sender (Rrep hops dip dsn oip sip) = sip"  
 " $\wedge$ dests sip. msg\_sender (Rerr dests sip) = sip"  
 " $\wedge$ d dip sip. msg\_sender (Pkt d dip sip) = sip"  
 unfolding msg\_sender\_def by simp\_all

definition msg\_zhops :: "msg  $\Rightarrow$  bool"

where "msg\_zhops m  $\equiv$  case m of  
 Rreq hopsc dipc \_ \_ oipc \_ sipc  $\Rightarrow$  hopsc = 0  $\longrightarrow$  oipc = sipc  
 | Rrep hopsc dipc \_ \_ sipc  $\Rightarrow$  hopsc = 0  $\longrightarrow$  dipc = sipc  
 | \_  $\Rightarrow$  True"

lemma msg\_zhops\_simps [simp]:

" $\wedge$ hops dip dsn dsk oip osn sip.  
 msg\_zhops (Rreq hops dip dsn dsk oip osn sip) = (hops = 0  $\longrightarrow$  oip = sip)"  
 " $\wedge$ hops dip dsn oip sip. msg\_zhops (Rrep hops dip dsn oip sip) = (hops = 0  $\longrightarrow$  dip = sip)"  
 " $\wedge$ dests sip. msg\_zhops (Rerr dests sip) = True"  
 " $\wedge$ d dip. msg\_zhops (Newpkt d dip) = True"  
 " $\wedge$ d dip sip. msg\_zhops (Pkt d dip sip) = True"  
 unfolding msg\_zhops\_def by simp\_all

definition rreq\_rrep\_sn :: "msg  $\Rightarrow$  bool"

where "rreq\_rrep\_sn m  $\equiv$  case m of Rreq \_ \_ \_ \_ \_ osnc \_  $\Rightarrow$  osnc  $\geq$  1  
 | Rrep \_ \_ dsnc \_ \_  $\Rightarrow$  dsnc  $\geq$  1  
 | \_  $\Rightarrow$  True"

lemma rreq\_rrep\_sn\_simps [simp]:

" $\wedge$ hops dip dsn dsk oip osn sip.  
 rreq\_rrep\_sn (Rreq hops dip dsn dsk oip osn sip) = (osn  $\geq$  1)"  
 " $\wedge$ hops dip dsn oip sip. rreq\_rrep\_sn (Rrep hops dip dsn oip sip) = (dsn  $\geq$  1)"  
 " $\wedge$ dests sip. rreq\_rrep\_sn (Rerr dests sip) = True"  
 " $\wedge$ d dip. rreq\_rrep\_sn (Newpkt d dip) = True"  
 " $\wedge$ d dip sip. rreq\_rrep\_sn (Pkt d dip sip) = True"  
 unfolding rreq\_rrep\_sn\_def by simp\_all

definition rreq\_rrep\_fresh :: "rt  $\Rightarrow$  msg  $\Rightarrow$  bool"

where "rreq\_rrep\_fresh crt m  $\equiv$  case m of Rreq hopsc \_ \_ \_ oipc osnc ipcc  $\Rightarrow$  (ipcc  $\neq$  oipc  $\longrightarrow$   
 oipc  $\in$  kD(crt)  $\wedge$  (sqn crt oipc > osnc  
 $\vee$  (sqn crt oipc = osnc  
 $\wedge$  the (dhops crt oipc)  $\leq$  hopsc  
 $\wedge$  the (flag crt oipc) = val)))  
 | Rreq hopsc dipc dsnc \_ ipcc  $\Rightarrow$  (ipcc  $\neq$  dipc  $\longrightarrow$   
 dipc  $\in$  kD(crt)  
 $\wedge$  sqn crt dipc = dsnc  
 $\wedge$  the (dhops crt dipc) = hopsc  
 $\wedge$  the (flag crt dipc) = val)  
 | \_  $\Rightarrow$  True"

lemma rreq\_rrep\_fresh\_simps [simp]:

" $\wedge$ hops dip dsn dsk oip osn sip.  
 rreq\_rrep\_fresh crt (Rreq hops dip dsn dsk oip osn sip) =  
 (sip  $\neq$  oip  $\longrightarrow$  oip  $\in$  kD(crt)  
 $\wedge$  (sqn crt oip > osn  
 $\vee$  (sqn crt oip = osn  
 $\wedge$  the (dhops crt oip)  $\leq$  hops  
 $\wedge$  the (flag crt oip) = val)))"  
 " $\wedge$ hops dip dsn oip sip. rreq\_rrep\_fresh crt (Rrep hops dip dsn oip sip) =

```

      (sip ≠ dip → dip ∈ kD(crt)
        ∧ sqn crt dip = dsn
        ∧ the (dhops crt dip) = hops
        ∧ the (flag crt dip) = val)"
"∧ dests sip.      rreq_rrep_fresh crt (Rerr dests sip) = True"
"∧ d dip.          rreq_rrep_fresh crt (Newpkt d dip) = True"
"∧ d dip sip.     rreq_rrep_fresh crt (Pkt d dip sip) = True"
unfolding rreq_rrep_fresh_def by simp_all

```

**definition** `rerr_invalid :: "rt ⇒ msg ⇒ bool"`

where `"rerr_invalid crt m ≡ case m of Rerr destsc _ ⇒ (∀ ripc ∈ dom(destsc).  
 (ripc ∈ iD(crt) ∧ the (destsc ripc) = sqn crt ripc))  
 | _ ⇒ True"`

**lemma** `rerr_invalid [simp]:`

```

"∧ hops dip dsn dsk oip osn sip.
  rerr_invalid crt (Rreq hops dip dsn dsk oip osn sip) = True"
"∧ hops dip dsn oip sip. rerr_invalid crt (Rrep hops dip dsn oip sip) = True"
"∧ dests sip.          rerr_invalid crt (Rerr dests sip) = (∀ ripc ∈ dom(dests).
  ripc ∈ iD(crt) ∧ the (dests ripc) = sqn crt ripc)"
"∧ d dip.              rerr_invalid crt (Newpkt d dip) = True"
"∧ d dip sip.         rerr_invalid crt (Pkt d dip sip) = True"
unfolding rerr_invalid_def by simp_all

```

**definition**

`initmissing :: "(nat ⇒ state option) × 'a ⇒ (nat ⇒ state) × 'a"`

where

`"initmissing σ = (λi. case (fst σ) i of None ⇒ aadv_init i | Some s ⇒ s, snd σ)"`

**lemma** `not_in_net_ips_fst_init_missing [simp]:`

```

assumes "i ∉ net_ips σ"
shows "fst (initmissing (netgmap fst σ)) i = aadv_init i"
using assms unfolding initmissing_def by simp

```

**lemma** `fst_initmissing_netgmap_pair_fst [simp]:`

```

"fst (initmissing (netgmap (λ(p, q). (fst (id p), snd (id p), q)) s))
  = fst (initmissing (netgmap fst s))"
unfolding initmissing_def by auto

```

We introduce a streamlined alternative to `initmissing` with `netgmap` to simplify invariant statements and thus facilitate their comprehension and presentation.

**lemma** `fst_initmissing_netgmap_default_aadv_init_netlift:`

```

"fst (initmissing (netgmap fst s)) = default aadv_init (netlift fst s)"
unfolding initmissing_def default_def
by (simp add: fst_netgmap_netlift del: One_nat_def)

```

**definition**

`netglobal :: "(nat ⇒ state) ⇒ bool ⇒ ((state × 'b) × 'c) net_state ⇒ bool"`

where

`"netglobal P ≡ (λs. P (default aadv_init (netlift fst s)))"`

end

## 1.5 Quality relations between routes

theory `A_Fresher`

imports `A_Aadv_Data`

begin

### 1.5.1 Net sequence numbers

On individual routes

**definition**

```

  nsqnr :: "r ⇒ sqn"
where
  "nsqnr r ≡ if π4(r) = val ∨ π2(r) = 0 then π2(r) else (π2(r) - 1)"

lemma nsqnr_def':
  "nsqnr r = (if π4(r) = inv then π2(r) - 1 else π2(r))"
  unfolding nsqnr_def by simp

lemma nsqnr_zero [simp]:
  "∧ dsn dsk flag hops nhip pre. nsqnr (0, dsk, flag, hops, nhip, pre) = 0"
  unfolding nsqnr_def by clarsimp

lemma nsqnr_val [simp]:
  "∧ dsn dsk hops nhip pre. nsqnr (dsn, dsk, val, hops, nhip, pre) = dsn"
  unfolding nsqnr_def by clarsimp

lemma nsqnr_inv [simp]:
  "∧ dsn dsk hops nhip pre. nsqnr (dsn, dsk, inv, hops, nhip, pre) = dsn - 1"
  unfolding nsqnr_def by clarsimp

lemma nsqnr_lte_dsn [simp]:
  "∧ dsn dsk flag hops nhip pre. nsqnr (dsn, dsk, flag, hops, nhip, pre) ≤ dsn"
  unfolding nsqnr_def by clarsimp

```

## On routes in routing tables

### definition

```

  nsqn :: "rt ⇒ ip ⇒ sqn"
where
  "nsqn ≡ λrt dip. case σroute(rt, dip) of None ⇒ 0 | Some r ⇒ nsqnr(r)"

```

### lemma nsqn\_sqn\_def:

```

  "∧rt dip. nsqn rt dip = (if flag rt dip = Some val ∨ sqn rt dip = 0
    then sqn rt dip else sqn rt dip - 1)"
  unfolding nsqn_def sqn_def by (clarsimp split: option.split) auto

```

### lemma not\_in\_kD\_nsqn [simp]:

```

  assumes "dip ∉ kD(rt)"
  shows "nsqn rt dip = 0"
  using assms unfolding nsqn_def by simp

```

### lemma kD\_nsqn:

```

  assumes "dip ∈ kD(rt)"
  shows "nsqn rt dip = nsqnr(the (σroute(rt, dip)))"
  using assms [THEN kD_Some] unfolding nsqn_def by clarsimp

```

### lemma nsqnr\_r\_flag\_pred [simp, intro]:

```

  fixes dsn dsk flag hops nhip pre
  assumes "P (nsqnr (dsn, dsk, val, hops, nhip, pre))"
  and "P (nsqnr (dsn, dsk, inv, hops, nhip, pre))"
  shows "P (nsqnr (dsn, dsk, flag, hops, nhip, pre))"
  using assms by (cases flag) auto

```

### lemma nsqn<sub>r</sub>\_addpreRT\_inv [simp]:

```

  "∧rt dip npre dip'. dip ∈ kD(rt) ⇒
  nsqnr (the (the (addpreRT rt dip npre) dip')) = nsqnr (the (rt dip'))"
  unfolding addpreRT_def nsqnr_def
  by (frule kD_Some) (clarsimp split: option.split)

```

### lemma sqn\_nsqn:

```

  "∧rt dip. sqn rt dip - 1 ≤ nsqn rt dip"
  unfolding sqn_def nsqn_def by (clarsimp split: option.split)

```

### lemma nsqn\_sqn: "nsqn rt dip ≤ sqn rt dip"

```

unfolding sqn_def nsqn_def by (cases "rt dip") auto

lemma val_nsqn_sqn [elim, simp]:
  assumes "ip ∈ kD(rt)"
    and "the (flag rt ip) = val"
  shows "nsqn rt ip = sqn rt ip"
  using assms unfolding nsqn_sqn_def by auto

lemma vD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ vD(rt)"
  shows "nsqn rt ip = sqn rt ip"
  proof -
    from ⟨ip ∈ vD(rt)⟩ have "ip ∈ kD(rt)"
      and "the (flag rt ip) = val" by auto
    thus ?thesis ..
  qed

lemma inv_nsqn_sqn [elim, simp]:
  assumes "ip ∈ kD(rt)"
    and "the (flag rt ip) = inv"
  shows "nsqn rt ip = sqn rt ip - 1"
  using assms unfolding nsqn_sqn_def by auto

lemma iD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ iD(rt)"
  shows "nsqn rt ip = sqn rt ip - 1"
  proof -
    from ⟨ip ∈ iD(rt)⟩ have "ip ∈ kD(rt)"
      and "the (flag rt ip) = inv" by auto
    thus ?thesis ..
  qed

lemma nsqn_update_changed_kno_val [simp]: "∧rt ip dsn dsk hops nhip.
  rt ≠ update rt ip (dsn, kno, val, hops, nhip, { })
  ⇒ nsqn (update rt ip (dsn, kno, val, hops, nhip, { })) ip = dsn"
  using assms unfolding nsqn_r_def update_def
  by (clarsimp simp: kD_nsqn split: option.split_asm option.split split_if_asm)
  (metis fun_upd_triv)

lemma nsqn_addpreRT_inv [simp]:
  "∧rt dip npre dip'. dip ∈ kD(rt) ⇒
  nsqn (the (addpreRT rt dip npre)) dip' = nsqn rt dip'"
  unfolding addpreRT_def nsqn_def nsqn_r_def
  by (frule kD_Some) (clarsimp split: option.split)

lemma nsqn_update_other [simp]:
  fixes dsn dsk flag hops dip nhip pre rt ip
  assumes "dip ≠ ip"
  shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip, pre)) dip = nsqn rt dip"
  using assms unfolding nsqn_def
  by (clarsimp split: option.split)

lemma nsqn_invalidate_eq:
  assumes "dip ∈ kD(rt)"
    and "dests dip = Some rsn"
  shows "nsqn (invalidate rt dests) dip = rsn - 1"
  using assms
  proof -
    from assms obtain dsk hops nhip pre
      where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip, pre)"
      unfolding invalidate_def
      by auto
    moreover from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp
    ultimately show ?thesis
  qed

```

```

    using ⟨dests dip = Some rsn⟩ by simp
qed

```

```

lemma nsqn_invalidate_other [simp]:
  assumes "dip ∈ kD(rt)"
    and "dip ∉ dom dests"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
using assms by (clarsimp simp add: kD_nsqn)

```

## 1.5.2 Comparing routes

definition

```

fresher :: "r ⇒ r ⇒ bool" ("(_/ ⊆ _)" [51, 51] 50)

```

where

```

"fresher r r' ≡ ((nsqnr r < nsqnr r') ∨ (nsqnr r = nsqnr r' ∧ π5(r) ≥ π5(r')))"

```

```

lemma fresherI1 [intro]:
  assumes "nsqnr r < nsqnr r'"
  shows "r ⊆ r'"
unfolding fresher_def using assms by simp

```

```

lemma fresherI2 [intro]:
  assumes "nsqnr r = nsqnr r'"
    and "π5(r) ≥ π5(r')"
  shows "r ⊆ r'"
unfolding fresher_def using assms by simp

```

```

lemma fresherI [intro]:
  assumes "(nsqnr r < nsqnr r') ∨ (nsqnr r = nsqnr r' ∧ π5(r) ≥ π5(r'))"
  shows "r ⊆ r'"
unfolding fresher_def using assms .

```

```

lemma fresherE [elim]:
  assumes "r ⊆ r'"
    and "nsqnr r < nsqnr r' ⇒ P r r'"
    and "nsqnr r = nsqnr r' ∧ π5(r) ≥ π5(r') ⇒ P r r'"
  shows "P r r'"
using assms unfolding fresher_def by auto

```

```

lemma fresher_refl [simp]: "r ⊆ r"
unfolding fresher_def by simp

```

```

lemma fresher_trans [elim, trans]:
  "[ x ⊆ y; y ⊆ z ] ⇒ x ⊆ z"
unfolding fresher_def by auto

```

```

lemma not_fresher_trans [elim, trans]:
  "[ ¬(x ⊆ y); ¬(z ⊆ x) ] ⇒ ¬(z ⊆ y)"
unfolding fresher_def by auto

```

```

lemma fresher_dsn_flag_hops_const [simp]:
  fixes dsn dsk dsk' flag hops nhip nhip' pre pre'
  shows "(dsn, dsk, flag, hops, nhip, pre) ⊆ (dsn, dsk', flag, hops, nhip', pre')"
unfolding fresher_def by (cases flag) simp_all

```

```

lemma addpre_fresher [simp]: "∧r npre. r ⊆ (addpre r npre)"
by clarsimp

```

## 1.5.3 Comparing routing tables

definition

```

rt_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"

```

where

```

"rt_fresher ≡ λdip rt rt'. (the (σroute(rt, dip))) ⊆ (the (σroute(rt', dip)))"

```



abbreviation

`rt_fresher_syn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  rt  $\Rightarrow$  bool" ("(_/  $\sqsubseteq$  _)" [51, 999, 51] 50)`

where

`"rt1  $\sqsubseteq_i$  rt2  $\equiv$  rt_fresher i rt1 rt2"`

lemma `rt_fresher_def'`:

`"(rt1  $\sqsubseteq_i$  rt2) = (nsqn_r (the (rt1 i)) < nsqn_r (the (rt2 i))  $\vee$   
nsqn_r (the (rt1 i)) = nsqn_r (the (rt2 i))  $\wedge$   $\pi_5$  (the (rt2 i))  $\leq$   $\pi_5$  (the (rt1 i)))"`  
unfolding `rt_fresher_def` `fresher_def` by (rule `refl`)

lemma `single_rt_fresher` [intro]:

assumes `"the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"`  
shows `"rt1  $\sqsubseteq_{ip}$  rt2"`  
using `assms` unfolding `rt_fresher_def` .

lemma `rt_fresher_single` [intro]:

assumes `"rt1  $\sqsubseteq_{ip}$  rt2"`  
shows `"the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"`  
using `assms` unfolding `rt_fresher_def` .

lemma `rt_fresher_def2`:

assumes `"dip  $\in$  kD(rt1)"`  
and `"dip  $\in$  kD(rt2)"`  
shows `"(rt1  $\sqsubseteq_{dip}$  rt2) = (nsqn rt1 dip < nsqn rt2 dip  
 $\vee$  (nsqn rt1 dip = nsqn rt2 dip  
 $\wedge$  the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip)))"`  
using `assms` unfolding `rt_fresher_def` `fresher_def` by (simp add: `kD_nsqn proj5_eq_dhops`)

lemma `rt_fresherI1` [intro]:

assumes `"dip  $\in$  kD(rt1)"`  
and `"dip  $\in$  kD(rt2)"`  
and `"nsqn rt1 dip < nsqn rt2 dip"`  
shows `"rt1  $\sqsubseteq_{dip}$  rt2"`  
unfolding `rt_fresher_def2` [OF `assms(1-2)`] using `assms(3)` by `simp`

lemma `rt_fresherI2` [intro]:

assumes `"dip  $\in$  kD(rt1)"`  
and `"dip  $\in$  kD(rt2)"`  
and `"nsqn rt1 dip = nsqn rt2 dip"`  
and `"the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip)"`  
shows `"rt1  $\sqsubseteq_{dip}$  rt2"`  
unfolding `rt_fresher_def2` [OF `assms(1-2)`] using `assms(3-4)` by `simp`

lemma `rt_fresherE` [elim]:

assumes `"rt1  $\sqsubseteq_{dip}$  rt2"`  
and `"dip  $\in$  kD(rt1)"`  
and `"dip  $\in$  kD(rt2)"`  
and `"[ nsqn rt1 dip < nsqn rt2 dip ]  $\implies$  P rt1 rt2 dip"`  
and `"[ nsqn rt1 dip = nsqn rt2 dip;  
the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip) ]  $\implies$  P rt1 rt2 dip"`  
shows `"P rt1 rt2 dip"`  
using `assms(1)` unfolding `rt_fresher_def2` [OF `assms(2-3)`]  
using `assms(4-5)` by `auto`

lemma `rt_fresher_refl` [simp]: `"rt  $\sqsubseteq_{dip}$  rt"`

unfolding `rt_fresher_def` by `simp`

lemma `rt_fresher_trans` [elim, trans]:

assumes `"rt1  $\sqsubseteq_{dip}$  rt2"`  
and `"rt2  $\sqsubseteq_{dip}$  rt3"`  
shows `"rt1  $\sqsubseteq_{dip}$  rt3"`  
using `assms` unfolding `rt_fresher_def` by `auto`

```

lemma rt_fresher_if_Some [intro!]:
  assumes "the (rt dip)  $\sqsubseteq$  r"
  shows "rt  $\sqsubseteq_{dip}$  ( $\lambda ip. \text{if } ip = \text{dip then Some } r \text{ else rt } ip$ )"
  using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip  $\Rightarrow$  rt  $\Rightarrow$  rt  $\Rightarrow$  bool"
where
  "rt_fresh_as  $\equiv \lambda dip \text{ rt1 rt2. } (rt1 \sqsubseteq_{dip} rt2) \wedge (rt2 \sqsubseteq_{dip} rt1)"$ 

abbreviation
  rt_fresh_as_syn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  rt  $\Rightarrow$  bool" ("(_ /  $\approx$  _)" [51, 999, 51] 50)
where
  "rt1  $\approx_i$  rt2  $\equiv$  rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_refl [simp]: " $\wedge$ rt dip. rt  $\approx_{dip}$  rt"
  unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
  " $\wedge$ rt1 rt2 rt3 dip. [ $rt1 \approx_{dip} rt2$ ;  $rt2 \approx_{dip} rt3$ ]  $\implies$  rt1  $\approx_{dip}$  rt3"
  unfolding rt_fresh_as_def rt_fresher_def
  by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  and "rt2  $\sqsubseteq_{dip}$  rt1"
  shows "rt1  $\approx_{dip}$  rt2"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
  assumes "dip  $\in$  kD(rt1)"
  and "dip  $\in$  kD(rt2)"
  and "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"
  and "the (rt2 dip)  $\sqsubseteq$  the (rt1 dip)"
  shows "rt1  $\approx_{dip}$  rt2"
  using assms unfolding rt_fresh_as_def
  by (clarsimp dest!: single_rt_fresher)

lemma nsqn_rt_fresh_asI:
  assumes "dip  $\in$  kD(rt)"
  and "dip  $\in$  kD(rt')"
  and "nsqn rt dip = nsqn rt' dip"
  and " $\pi_5$ (the (rt dip)) =  $\pi_5$ (the (rt' dip))"
  shows "rt  $\approx_{dip}$  rt'"
proof
  from assms(1-2,4) have dhops': "the (dhops rt' dip)  $\leq$  the (dhops rt dip)"
  by (simp add: proj5_eq_dhops)
  with assms(1-3) show "rt  $\sqsubseteq_{dip}$  rt'"
  by (rule rt_fresherI2)
next
  from assms(1-2,4) have dhops: "the (dhops rt dip)  $\leq$  the (dhops rt' dip)"
  by (simp add: proj5_eq_dhops)
  with assms(2,1) assms(3) [symmetric] show "rt'  $\sqsubseteq_{dip}$  rt"
  by (rule rt_fresherI2)
qed

lemma rt_fresh_asE [elim]:
  assumes "rt1  $\approx_{dip}$  rt2"
  and "[ $rt1 \sqsubseteq_{dip} rt2$ ;  $rt2 \sqsubseteq_{dip} rt1$ ]  $\implies$  P rt1 rt2 dip"
  shows "P rt1 rt2 dip"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD1 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"

```

```

using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\sqsubseteq_{dip}$  rt1"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\approx_{dip}$  rt1"
  using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
  assumes " $\neg$  (rt1  $\sqsubseteq_{dip}$  rt2)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
  proof
    assume "rt1  $\approx_{dip}$  rt2"
    hence "rt1  $\sqsubseteq_{dip}$  rt2" ..
    with ( $\neg$  (rt1  $\sqsubseteq_{dip}$  rt2)) show False ..
  qed

lemma not_rt_fresh_asI2 [intro]:
  assumes " $\neg$  (rt2  $\sqsubseteq_{dip}$  rt1)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
  proof
    assume "rt1  $\approx_{dip}$  rt2"
    hence "rt2  $\sqsubseteq_{dip}$  rt1" ..
    with ( $\neg$  (rt2  $\sqsubseteq_{dip}$  rt1)) show False ..
  qed

lemma not_single_rt_fresher [elim]:
  assumes " $\neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))"
  shows " $\neg$ (rt1  $\sqsubseteq_{ip}$  rt2)"
  proof
    assume "rt1  $\sqsubseteq_{ip}$  rt2"
    hence "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)" ..
    with ( $\neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))) show False ..
  qed

lemmas not_single_rt_fresh_asI1 [intro] = not_rt_fresh_asI1 [OF not_single_rt_fresher]
lemmas not_single_rt_fresh_asI2 [intro] = not_rt_fresh_asI2 [OF not_single_rt_fresher]

lemma not_rt_fresher_single [elim]:
  assumes " $\neg$ (rt1  $\sqsubseteq_{ip}$  rt2)"
  shows " $\neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))"
  proof
    assume "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"
    hence "rt1  $\sqsubseteq_{ip}$  rt2" ..
    with ( $\neg$ (rt1  $\sqsubseteq_{ip}$  rt2)) show False ..
  qed

lemma rt_fresh_as_nsqr:
  assumes "dip  $\in$  kD(rt1)"
  and "dip  $\in$  kD(rt2)"
  and "rt1  $\approx_{dip}$  rt2"
  shows "nsqnr (the (rt2 dip)) = nsqnr (the (rt1 dip))"
  using assms(3) unfolding rt_fresh_as_def
  by (auto simp: rt_fresher_def2 [OF <dip  $\in$  kD(rt1)> <dip  $\in$  kD(rt2)>]
    rt_fresher_def2 [OF <dip  $\in$  kD(rt2)> <dip  $\in$  kD(rt1)>]
    kD_nsqn [OF <dip  $\in$  kD(rt1)>]
    kD_nsqn [OF <dip  $\in$  kD(rt2)>])

lemma rt_fresher_mapupd [intro!]:
  assumes "dip  $\in$  kD(rt)"

```

```

    and "the (rt dip)  $\sqsubseteq$  r"
    shows "rt  $\sqsubseteq_{dip}$  rt(dip  $\mapsto$  r)"
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_map_update_other [intro!]:
  assumes "dip  $\in$  kD(rt)"
    and "dip  $\neq$  ip"
  shows "rt  $\sqsubseteq_{dip}$  rt(ip  $\mapsto$  r)"
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_update_other [simp]:
  assumes inkD: "dip  $\in$  kD(rt)"
    and "dip  $\neq$  ip"
  shows "rt  $\sqsubseteq_{dip}$  update rt ip r"
using assms unfolding update_def
by (clarsimp split: option.split) (fastforce)

theorem rt_fresher_update [simp]:
  assumes "dip  $\in$  kD(rt)"
    and "the (dhops rt dip)  $\geq$  1"
    and "update_arg_wf r"
  shows "rt  $\sqsubseteq_{dip}$  update rt ip r"
proof (cases "dip = ip")
  assume "dip  $\neq$  ip" with  $\langle$ dip  $\in$  kD(rt) $\rangle$  show ?thesis
    by (rule rt_fresher_update_other)
next
  assume "dip = ip"

  from  $\langle$ dip  $\in$  kD(rt) $\rangle$  obtain dsnn dskn fn hopsn nhipn pren
    where rtn [simp]: "the (rt dip) = (dsnn, dskn, fn, hopsn, nhipn, pren)"
    by (metis prod_cases6)
  with  $\langle$ the (dhops rt dip)  $\geq$  1 $\rangle$  and  $\langle$ dip  $\in$  kD(rt) $\rangle$  have "hopsn  $\geq$  1"
    by (metis proj5_eq_dhops projs(4))
  from  $\langle$ dip  $\in$  kD(rt) $\rangle$  rtn have [simp]: "sqn rt dip = dsnn"
    and [simp]: "the (dhops rt dip) = hopsn"
    and [simp]: "the (flag rt dip) = fn"
  by (simp add: sqn_def proj5_eq_dhops [symmetric]
    proj4_eq_flag [symmetric])

  from  $\langle$ update_arg_wf r $\rangle$  have "(dsnn, dskn, fn, hopsn, nhipn, pren)
     $\sqsubseteq$  the ((update rt dip r) dip)"
  proof (rule wf_r_cases)
    fix nhip pre
    from  $\langle$ hopsn  $\geq$  1 $\rangle$  have " $\wedge$ pre'. (dsnn, dskn, fn, hopsn, nhipn, pren)
       $\sqsubseteq$  (dsnn, unk, val, Suc 0, nhip, pre')"
    unfolding fresher_def sqn_def by (cases fn) auto
  thus "(dsnn, dskn, fn, hopsn, nhipn, pren)
     $\sqsubseteq$  the (update rt dip (0, unk, val, Suc 0, nhip, pre) dip)"
    using  $\langle$ dip  $\in$  kD(rt) $\rangle$  by - (rule update_cases_kD, simp_all)
  next
    fix dsn :: sqn and hops nhip pre
    assume "0 < dsn"
    show "(dsnn, dskn, fn, hopsn, nhipn, pren)
       $\sqsubseteq$  the (update rt dip (dsn, kno, val, hops, nhip, pre) dip)"
    proof (rule update_cases_kD [OF _  $\langle$ dip  $\in$  kD(rt) $\rangle$ ], simp_all add:  $\langle$ 0 < dsn $\rangle$ )
      assume "dsnn < dsn"
      thus "(dsnn, dskn, fn, hopsn, nhipn, pren)
         $\sqsubseteq$  (dsn, kno, val, hops, nhip, pre  $\cup$  pren)"
        unfolding fresher_def by auto
    next
      assume "dsnn = dsn"
      and "hops < hopsn"
      thus "(dsn, dskn, fn, hopsn, nhipn, pren)
         $\sqsubseteq$  (dsn, kno, val, hops, nhip, pre  $\cup$  pren)"

```

```

      unfolding fresher_def nsqn_r_def by simp
next
  assume "dsnn = dsn"
  with ⟨0 < dsn⟩
  show "(dsn, dskn, inv, hopsn, nhipn, pren)
    ⊆ (dsn, kno, val, hops, nhip, pre ∪ pren)"
  unfolding fresher_def by simp
qed
qed
hence "rt ⊆dip update rt dip r"
  by - (rule single_rt_fresher, simp)
with ⟨dip = ip⟩ show ?thesis by simp
qed

theorem rt_fresher_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
    and indests: "∀ rip ∈ dom(dests). rip ∈ vD(rt) ∧ sqn rt rip < the (dests rip)"
  shows "rt ⊆dip invalidate rt dests"
proof (cases "dip ∈ dom(dests)")
  assume "dip ∉ dom(dests)"
  thus ?thesis using ⟨dip ∈ kD(rt)⟩
  by - (rule single_rt_fresher, simp)
next
  assume "dip ∈ dom(dests)"
  moreover with indests have "dip ∈ vD(rt)"
    and "sqn rt dip < the (dests dip)"
  by auto
  ultimately show ?thesis
  unfolding invalidate_def sqn_def
  by - (rule single_rt_fresher, auto simp: fresher_def)
qed

lemma nsqn_r_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
    and "dip ∈ dom(dests)"
  shows "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
  using assms unfolding invalidate_def by auto

lemma rt_fresh_as_inc_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
    and "∀ rip ∈ dom(dests). rip ∈ vD(rt) ∧ the (dests rip) = inc (sqn rt rip)"
  shows "rt ≈dip invalidate rt dests"
proof (cases "dip ∈ dom(dests)")
  assume "dip ∉ dom(dests)"
  with ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)"
  by simp
  with ⟨dip ∈ kD(rt)⟩ show ?thesis
  by rule (simp_all add: ⟨dip ∉ dom(dests)⟩)
next
  assume "dip ∈ dom(dests)"
  with assms(2) have "dip ∈ vD(rt)"
    and "the (dests dip) = inc (sqn rt dip)" by auto
  from ⟨dip ∈ vD(rt)⟩ have "dip ∈ kD(rt)" by simp
  moreover then have "dip ∈ kD(invalidate rt dests)" by simp
  ultimately show ?thesis
proof (rule nsqn_rt_fresh_asI)
  from ⟨dip ∈ vD(rt)⟩ have "nsqn rt dip = sqn rt dip" by simp
  also have "sqn rt dip = nsqnr (the (invalidate rt dests dip))"
  proof -
  from ⟨dip ∈ kD(rt)⟩ have "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
    using ⟨dip ∈ dom(dests)⟩ by (rule nsqn_r_invalidate)
  with ⟨the (dests dip) = inc (sqn rt dip)⟩
  show "sqn rt dip = nsqnr (the (invalidate rt dests dip))" by simp
  qed
  qed
qed

```

```

also from ⟨dip∈kD(invalidate rt dests)⟩
  have "nsqnr (the (invalidate rt dests dip)) = nsqn (invalidate rt dests) dip"
    by (simp add: kD_nsqn)
  finally show "nsqn rt dip = nsqn (invalidate rt dests) dip" .
qed simp
qed

```

```
lemmas rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]
```

```
lemma rt_fresh_as_addpreRT [simp]:
  assumes "ip∈kD(rt)"
  shows "rt ≈dip the (addpreRT rt ip npre)"
  using assms [THEN kD_Some] by (auto simp: addpreRT_def)

```

```
lemmas rt_fresher_addpreRT [simp] = rt_fresh_as_addpreRT [THEN rt_fresh_asD1]
```

#### 1.5.4 Strictly comparing routing tables

```
definition rt_strictly_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"

```

where

```
"rt_strictly_fresher ≡ λdip rt1 rt2. (rt1 ⊆dip rt2) ∧ ¬(rt1 ≈dip rt2)"

```

abbreviation

```
rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊆ _)" [51, 999, 51] 50)

```

where

```
"rt1 ⊆i rt2 ≡ rt_strictly_fresher i rt1 rt2"

```

```
lemma rt_strictly_fresher_def'':
  "rt1 ⊆i rt2 = ((rt1 ⊆i rt2) ∧ ¬(rt2 ⊆i rt1))"
  unfolding rt_strictly_fresher_def rt_fresh_as_def by auto

```

```
lemma rt_strictly_fresherI' [intro]:
  assumes "rt1 ⊆i rt2"
  and "¬(rt2 ⊆i rt1)"
  shows "rt1 ⊆i rt2"
  using assms unfolding rt_strictly_fresher_def'' by simp

```

```
lemma rt_strictly_fresherE' [elim]:
  assumes "rt1 ⊆i rt2"
  and "[| rt1 ⊆i rt2; ¬(rt2 ⊆i rt1) |] ⇒ P rt1 rt2 i"
  shows "P rt1 rt2 i"
  using assms unfolding rt_strictly_fresher_def'' by simp

```

```
lemma rt_strictly_fresherI [intro]:
  assumes "rt1 ⊆i rt2"
  and "¬(rt1 ≈i rt2)"
  shows "rt1 ⊆i rt2"
  unfolding rt_strictly_fresher_def using assms ..

```

```
lemmas rt_strictly_fresher_singleI [elim] = rt_strictly_fresherI [OF single_rt_fresher]
```

```
lemma rt_strictly_fresherE [elim]:
  assumes "rt1 ⊆i rt2"
  and "[| rt1 ⊆i rt2; ¬(rt1 ≈i rt2) |] ⇒ P rt1 rt2 i"
  shows "P rt1 rt2 i"
  using assms(1) unfolding rt_strictly_fresher_def
  by rule (erule(1) assms(2))

```

```
lemma rt_strictly_fresher_def':
  "rt1 ⊆i rt2 =
  (nsqnr (the (rt1 i)) < nsqnr (the (rt2 i))
  ∨ (nsqnr (the (rt1 i)) = nsqnr (the (rt2 i)) ∧ π5(the (rt1 i)) > π5(the (rt2 i))))"
  unfolding rt_strictly_fresher_def'' rt_fresher_def fresher_def by auto

```

```

lemma rt_strictly_fresher_fresherD [dest]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    shows "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"
  using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    shows " $\neg$  rt1  $\approx_{dip}$  rt2"
  using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
    shows "rt1  $\sqsubseteq_{dip}$  rt3"
  using assms proof -
    from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  obtain "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)" by auto
    also from  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  obtain "the (rt2 dip)  $\sqsubseteq$  the (rt3 dip)" by auto
    finally have "the (rt1 dip)  $\sqsubseteq$  the (rt3 dip)" .

    moreover have " $\neg$  (rt1  $\approx_{dip}$  rt3)"
  proof -
    from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  obtain " $\neg$ (the (rt2 dip)  $\sqsubseteq$  the (rt1 dip))" by auto
    also from  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  obtain " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt2 dip))" by auto
    finally have " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt1 dip))" .
    thus ?thesis ..
  qed
  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: " $\neg$  (rt  $\sqsubseteq_{dip}$  rt)"
  using assms unfolding rt_strictly_fresher_def
  by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
    shows "rt1  $\sqsubseteq_{dip}$  rt3"
  proof -
    from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  have "rt1  $\sqsubseteq_{dip}$  rt2"
      and " $\neg$ (rt2  $\sqsubseteq_{dip}$  rt1)"
    unfolding rt_strictly_fresher_def'' by auto
    from this(1) and  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  have "rt1  $\sqsubseteq_{dip}$  rt3" ..

    moreover from  $\langle$  $\neg$ (rt2  $\sqsubseteq_{dip}$  rt1) $\rangle$  have " $\neg$ (rt3  $\sqsubseteq_{dip}$  rt1)"
  proof (rule contrapos_nn)
    assume "rt3  $\sqsubseteq_{dip}$  rt1"
    with  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  show "rt2  $\sqsubseteq_{dip}$  rt1" ..
  qed

  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3"
    unfolding rt_strictly_fresher_def'' by auto
qed

lemma rt_fresher_trans_rt_strictly_fresher' [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
    shows "rt1  $\sqsubseteq_{dip}$  rt3"
  proof -
    from  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  have "rt2  $\sqsubseteq_{dip}$  rt3"
      and " $\neg$ (rt3  $\sqsubseteq_{dip}$  rt2)"
    unfolding rt_strictly_fresher_def'' by auto
    from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  and this(1) have "rt1  $\sqsubseteq_{dip}$  rt3" ..

    moreover from  $\langle$  $\neg$ (rt3  $\sqsubseteq_{dip}$  rt2) $\rangle$  have " $\neg$ (rt3  $\sqsubseteq_{dip}$  rt1)"

```

```

proof (rule contrapos_nn)
  assume "rt3  $\sqsubseteq_{dip}$  rt1"
  thus "rt3  $\sqsubseteq_{dip}$  rt2" using ⟨rt1  $\sqsubseteq_{dip}$  rt2⟩ ..
qed

ultimately show "rt1  $\sqsubseteq_{dip}$  rt3"
  unfolding rt_strictly_fresher_def'' by auto
qed

lemma rt_fresher_imp_nsqn_le:
  assumes "rt1  $\sqsubseteq_{ip}$  rt2"
    and "ip  $\in$  kD rt1"
    and "ip  $\in$  kD rt2"
  shows "nsqn rt1 ip  $\leq$  nsqn rt2 ip"
using assms(1)
by (auto simp add: rt_fresher_def2 [OF assms(2-3)])

lemma rt_strictly_fresher_ltI [intro]:
  assumes "dip  $\in$  kD(rt1)"
    and "dip  $\in$  kD(rt2)"
    and "nsqn rt1 dip < nsqn rt2 dip"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
proof
  from assms show "rt1  $\sqsubseteq_{dip}$  rt2" ..
next
  show " $\neg$ (rt1  $\approx_{dip}$  rt2)"
  proof
    assume "rt1  $\approx_{dip}$  rt2"
    hence "rt2  $\sqsubseteq_{dip}$  rt1" ..
    hence "nsqn rt2 dip  $\leq$  nsqn rt1 dip"
      using ⟨dip  $\in$  kD(rt2)⟩ ⟨dip  $\in$  kD(rt1)⟩
      by (rule rt_fresher_imp_nsqn_le)
    with ⟨nsqn rt1 dip < nsqn rt2 dip⟩ show "False"
    by simp
  qed
qed

lemma rt_strictly_fresher_eqI [intro]:
  assumes "i  $\in$  kD(rt1)"
    and "i  $\in$  kD(rt2)"
    and "nsqn rt1 i = nsqn rt2 i"
    and " $\pi_5$ (the (rt2 i)) <  $\pi_5$ (the (rt1 i))"
  shows "rt1  $\sqsubseteq_i$  rt2"
using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

lemma invalidate_rtsf_left [simp]:
  " $\bigwedge$ dests dip rt rt'. dests dip = None  $\implies$  (invalidate rt dests  $\sqsubseteq_{dip}$  rt') = (rt  $\sqsubseteq_{dip}$  rt')"
  unfolding invalidate_def rt_strictly_fresher_def'
  by (rule iffI) (auto split: option.split_asm)

lemma vD_invalidate_rt_strictly_fresher [simp]:
  assumes "dip  $\in$  vD(invalidate rt1 dests)"
  shows "(invalidate rt1 dests  $\sqsubseteq_{dip}$  rt2) = (rt1  $\sqsubseteq_{dip}$  rt2)"
proof (cases "dip  $\in$  dom(dests)")
  assume "dip  $\in$  dom(dests)"
  hence "dip  $\notin$  vD(invalidate rt1 dests)"
  unfolding invalidate_def vD_def
  by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
  with ⟨dip  $\in$  vD(invalidate rt1 dests)⟩ show ?thesis by simp
next
  assume "dip  $\notin$  dom(dests)"
  hence "dests dip = None" by auto
  moreover with ⟨dip  $\in$  vD(invalidate rt1 dests)⟩ have "dip  $\in$  vD(rt1)"
  unfolding invalidate_def vD_def

```



```

    by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
ultimately show ?thesis
  unfolding invalidate_def rt_strictly_fresher_def' by auto
qed

lemma rt_strictly_fresher_update_other [elim!]:
  "\^dip ip rt r rt'. [| dip ≠ ip; rt ⊆dip rt' |] ⇒ update rt ip r ⊆dip rt'"
  unfolding rt_strictly_fresher_def' by clarsimp

lemma addpreRT_strictly_fresher [simp]:
  assumes "dip ∈ kD(rt)"
  shows "(the (addpreRT rt dip npre) ⊆ip rt2) = (rt ⊆ip rt2)"
  using assms unfolding rt_strictly_fresher_def' by clarsimp

lemma lt_sqn_imp_update_strictly_fresher:
  assumes "dip ∈ vD (rt2 nhip)"
  and *: "osn < sqn (rt2 nhip) dip"
  and **: "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
  shows "update rt dip (osn, kno, val, hops, nhip, {}) ⊆dip rt2 nhip"
  unfolding rt_strictly_fresher_def'
  proof (rule disjI1)
    from ** have "nsqn (update rt dip (osn, kno, val, hops, nhip, {})) dip = osn"
      by (rule nsqn_update_changed_kno_val)
    with ⟨dip ∈ vD (rt2 nhip)⟩
      have "nsqnr (the (update rt dip (osn, kno, val, hops, nhip, {}) dip)) = osn"
        by (simp add: kD_nsqn)
    also have "osn < sqn (rt2 nhip) dip" by (rule *)
    also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip dip))"
      unfolding nsqnr_def using ⟨dip ∈ vD (rt2 nhip)⟩
      by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
    finally show "nsqnr (the (update rt dip (osn, kno, val, hops, nhip, {}) dip))
      < nsqnr (the (rt2 nhip dip))" .
  qed

lemma dhops_le_hops_imp_update_strictly_fresher:
  assumes "dip ∈ vD (rt2 nhip)"
  and sqn: "sqn (rt2 nhip) dip = osn"
  and hop: "the (dhops (rt2 nhip) dip) ≤ hops"
  and **: "rt ≠ update rt dip (osn, kno, val, Suc hops, nhip, {})"
  shows "update rt dip (osn, kno, val, Suc hops, nhip, {}) ⊆dip rt2 nhip"
  unfolding rt_strictly_fresher_def'
  proof (rule disjI2, rule conjI)
    from ** have "nsqn (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip = osn"
      by (rule nsqn_update_changed_kno_val)
    with ⟨dip ∈ vD (rt2 nhip)⟩
      have "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip)) = osn"
        by (simp add: kD_nsqn)
    also have "osn = sqn (rt2 nhip) dip" by (rule sqn [symmetric])
    also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip dip))"
      unfolding nsqnr_def using ⟨dip ∈ vD (rt2 nhip)⟩
      by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
    finally show "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip))
      = nsqnr (the (rt2 nhip dip))" .
  next
    have "the (dhops (rt2 nhip) dip) ≤ hops" by (rule hop)
    also have "hops < hops + 1" by simp
    also have "hops + 1 = the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)"
      using ** by simp
    finally have "the (dhops (rt2 nhip) dip)
      < the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)" .
    thus "π5 (the (rt2 nhip dip)) < π5 (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip))"
      using ⟨dip ∈ vD (rt2 nhip)⟩ by (simp add: proj5_eq_dhops)
  qed

```

```

lemma nsqn_invalidate:
  assumes "dip ∈ kD(rt)"
    and "∀ip∈dom(dests). ip ∈ vD(rt) ∧ the (dests ip) = inc (sqn rt ip)"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
proof -
  from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp

  from assms have "rt ≈dip invalidate rt dests"
  by (rule rt_fresh_as_inc_invalidate)
with ⟨dip ∈ kD(rt)⟩ ⟨dip ∈ kD(invalidate rt dests)⟩ show ?thesis
  by (simp add: kD_nsqn del: invalidate_kD_inv)
  (erule(2) rt_fresh_as_nsqnr)
qed

end

```

## 1.6 Invariant proofs on individual processes

```

theory A_Seq_Invariants
imports "../.../AWN/Invariants" A_Aodv A_Aodv_Data A_Aodv_Predicates A_Fresher

begin

```

The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

```

lemma sequence_number_increases:
  "paodv i ⊨A onll ΓAODV (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
  by inv_cterms

```

```

lemma sequence_number_one_or_bigger:
  "paodv i ⊨ onl ΓAODV (λ(ξ, _). 1 ≤ sn ξ)"
  by (rule onll_step_to_invariantI [OF sequence_number_increases])
  (auto simp: σAODV_def)

```

We can get rid of the onl/onll if desired...

```

lemma sequence_number_increases':
  "paodv i ⊨A (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
  by (rule step_invariant_weakenE [OF sequence_number_increases]) (auto dest!: onllD)

```

```

lemma sequence_number_one_or_bigger':
  "paodv i ⊨ (λ(ξ, _). 1 ≤ sn ξ)"
  by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto

```

```

lemma sip_in_kD:
  "paodv i ⊨ onl ΓAODV (λ(ξ, l). l ∈ ({PAodv-:7} ∪ {PAodv-:5} ∪ {PRrep-:0..PRrep-:1}
    ∪ {PRreq-:0..PRreq-:3}) → sip ξ ∈ kD (rt ξ))"
  by inv_cterms

```

```

lemma rrep_1_update_changes:
  "paodv i ⊨ onl ΓAODV (λ(ξ, l). (l = PRrep-:1 →
    rt ξ ≠ update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {})))"
  by inv_cterms

```

```

lemma addpreRT_partly_welldefined:
  "paodv i ⊨
  onl ΓAODV (λ(ξ, l). (l ∈ {PRreq-:16..PRreq-:18} ∪ {PRrep-:2..PRrep-:6} → dip ξ ∈ kD (rt ξ))
    ∧ (l ∈ {PRreq-:3..PRreq-:17} → oip ξ ∈ kD (rt ξ)))"
  by inv_cterms

```

Proposition 7.38

```

lemma includes_nhip:
  "paodv i ⊨ onl ΓAODV (λ(ξ, l). ∀dip∈kD(rt ξ). the (nhop (rt ξ) dip)∈kD(rt ξ))"

```

proof -

```

{ fix ip and  $\xi \xi'$  :: state
  assume " $\forall \text{dip} \in \text{kD}(\text{rt } \xi). \text{the } (\text{nhop}(\text{rt } \xi) \text{ dip}) \in \text{kD}(\text{rt } \xi)$ "
  and " $\xi' = \xi(\text{rt} := \text{update}(\text{rt } \xi) \text{ ip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{ip}, \{\}) )$ "
  hence " $\forall \text{dip} \in \text{kD}(\text{rt } \xi).$ 
    the  $(\text{nhop}(\text{update}(\text{rt } \xi) \text{ ip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{ip}, \{\})) \text{ dip}) = \text{ip}$ 
     $\vee \text{the } (\text{nhop}(\text{update}(\text{rt } \xi) \text{ ip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{ip}, \{\})) \text{ dip}) \in \text{kD}(\text{rt } \xi)$ "
  by clarsimp (metis nhop_update_unk_val update_another)
} note one_hop = this
{ fix ip sip sn hops and  $\xi \xi'$  :: state
  assume " $\forall \text{dip} \in \text{kD}(\text{rt } \xi). \text{the } (\text{nhop}(\text{rt } \xi) \text{ dip}) \in \text{kD}(\text{rt } \xi)$ "
  and " $\xi' = \xi(\text{rt} := \text{update}(\text{rt } \xi) \text{ ip } (\text{sn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip}, \{\}) )$ "
  and " $\text{sip} \in \text{kD}(\text{rt } \xi)$ "
  hence " $(\text{the } (\text{nhop}(\text{update}(\text{rt } \xi) \text{ ip } (\text{sn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip}, \{\})) \text{ ip}) = \text{ip}$ 
     $\vee \text{the } (\text{nhop}(\text{update}(\text{rt } \xi) \text{ ip } (\text{sn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip}, \{\})) \text{ ip}) \in \text{kD}(\text{rt } \xi)$ 
     $\wedge (\forall \text{dip} \in \text{kD}(\text{rt } \xi).$ 
      the  $(\text{nhop}(\text{update}(\text{rt } \xi) \text{ ip } (\text{sn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip}, \{\})) \text{ dip}) = \text{ip}$ 
       $\vee \text{the } (\text{nhop}(\text{update}(\text{rt } \xi) \text{ ip } (\text{sn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip}, \{\})) \text{ dip}) \in \text{kD}(\text{rt } \xi))$ "
  by (metis kD_update_unchanged nhop_update_changed update_another)
} note nhop_is_sip = this
show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf sip_in_kD]
      onl_invariant_sterms [OF aadv_wf addpreRT_partly_welldefined]
      solve: one_hop nhop_is_sip)

```

qed

Proposition 7.22: needed in Proposition 7.4

lemma addpreRT\_welldefined:

```

"paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, l). (l \in \{\text{PRreq-:16..PRreq-:18}\} \longrightarrow \text{dip } \xi \in \text{kD}(\text{rt } \xi)) \wedge$ 
  ( $l = \text{PRreq-:17} \longrightarrow \text{oip } \xi \in \text{kD}(\text{rt } \xi)) \wedge$ 
  ( $l = \text{PRrep-:5} \longrightarrow \text{dip } \xi \in \text{kD}(\text{rt } \xi)) \wedge$ 
  ( $l = \text{PRrep-:6} \longrightarrow (\text{the } (\text{nhop}(\text{rt } \xi) (\text{dip } \xi))) \in \text{kD}(\text{rt } \xi)))$ "

```

(is " $\_ \models$  onl  $\Gamma_{AODV} ?P$ ")

unfolding invariant\_def

proof

```

fix s
assume "s  $\in$  reachable (paadv i) TT"
then obtain  $\xi p$  where "s = ( $\xi, p$ )"
  and " $(\xi, p) \in \text{reachable}(\text{paadv } i) \text{ TT}$ "
  by (metis PairE)
have "onl  $\Gamma_{AODV} ?P (\xi, p)$ "
proof (rule onlI)
  fix l
  assume " $l \in \text{labels } \Gamma_{AODV} p$ "
  with  $\langle (\xi, p) \in \text{reachable}(\text{paadv } i) \text{ TT} \rangle$ 
  have I1: " $l \in \{\text{PRreq-:16..PRreq-:18}\} \longrightarrow \text{dip } \xi \in \text{kD}(\text{rt } \xi)$ "
  and I2: " $l = \text{PRreq-:17} \longrightarrow \text{oip } \xi \in \text{kD}(\text{rt } \xi)$ "
  and I3: " $l \in \{\text{PRrep-:2..PRrep-:6}\} \longrightarrow \text{dip } \xi \in \text{kD}(\text{rt } \xi)$ "
  by (auto dest!: invariantD [OF addpreRT_partly_welldefined])
  moreover from  $\langle (\xi, p) \in \text{reachable}(\text{paadv } i) \text{ TT} \rangle \langle l \in \text{labels } \Gamma_{AODV} p \rangle$  and I3
  have " $l = \text{PRrep-:6} \longrightarrow (\text{the } (\text{nhop}(\text{rt } \xi) (\text{dip } \xi))) \in \text{kD}(\text{rt } \xi)$ "
  by (auto dest!: invariantD [OF includes_nhip])
  ultimately show "?P ( $\xi, l$ )"
  by simp

```

qed

with  $\langle s = (\xi, p) \rangle$  show "onl  $\Gamma_{AODV} ?P s$ "

by simp

qed

Proposition 7.4

lemma known\_destinations\_increase:

```

"paadv i  $\models_A$  onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \text{kD}(\text{rt } \xi) \subseteq \text{kD}(\text{rt } \xi'))$ "
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf addpreRT_welldefined]
      simp add: subset_insertI)

```

Proposition 7.5

lemma rreqs\_increase:

```
"paodv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_)).$  rreqs  $\xi \subseteq$  rreqs  $\xi'$ )"
by (inv_cterms simp add: subset_insertI)
```

lemma dests\_bigger\_than\_sqn:

```
"paodv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l).$   $l \in \{PAodv-:15..PAodv-:19\}$ 
 $\cup \{PPkt-:7..PPkt-:11\}$ 
 $\cup \{PRreq-:9..PRreq-:13\}$ 
 $\cup \{PRreq-:21..PRreq-:25\}$ 
 $\cup \{PRrep-:10..PRrep-:14\}$ 
 $\cup \{PRerr-:1..PRerr-:5\}$ 
 $\longrightarrow (\forall ip \in \text{dom}(\text{dests } \xi). ip \in kD(\text{rt } \xi) \wedge \text{sqn } (\text{rt } \xi) \text{ ip} \leq \text{the } (\text{dests } \xi \text{ ip}))$ )"
```

proof -

have sqninv:

```
" $\wedge$ dests rt rsn ip.
 $\llbracket \forall ip \in \text{dom}(\text{dests}). ip \in kD(\text{rt}) \wedge \text{sqn } \text{rt } \text{ip} \leq \text{the } (\text{dests } \text{ip}); \text{dests } \text{ip} = \text{Some } \text{rsn} \rrbracket$ 
 $\implies \text{sqn } (\text{invalidate } \text{rt } \text{dests}) \text{ ip} \leq \text{rsn}"$ 
by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto
```

have indests:

```
" $\wedge$ dests rt rsn ip.
 $\llbracket \forall ip \in \text{dom}(\text{dests}). ip \in kD(\text{rt}) \wedge \text{sqn } \text{rt } \text{ip} \leq \text{the } (\text{dests } \text{ip}); \text{dests } \text{ip} = \text{Some } \text{rsn} \rrbracket$ 
 $\implies ip \in kD(\text{rt}) \wedge \text{sqn } \text{rt } \text{ip} \leq \text{rsn}"$ 
by (metis domI option.sel)
```

show ?thesis

by inv\_cterms

```
(clarsimp split: split_if_asm option.split_asm
elim!: sqninv indests)+
```

qed

Proposition 7.6

lemma sqns\_increase:

```
"paodv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_)). \forall ip. \text{sqn } (\text{rt } \xi) \text{ ip} \leq \text{sqn } (\text{rt } \xi') \text{ ip})"$ 
```

proof -

{ fix  $\xi ::$  state

assume \*: " $\forall ip \in \text{dom}(\text{dests } \xi). ip \in kD(\text{rt } \xi) \wedge \text{sqn } (\text{rt } \xi) \text{ ip} \leq \text{the } (\text{dests } \xi \text{ ip})"$

have " $\forall ip. \text{sqn } (\text{rt } \xi) \text{ ip} \leq \text{sqn } (\text{invalidate } (\text{rt } \xi) (\text{dests } \xi)) \text{ ip}"$

proof

fix ip

from \* have " $ip \notin \text{dom}(\text{dests } \xi) \vee \text{sqn } (\text{rt } \xi) \text{ ip} \leq \text{the } (\text{dests } \xi \text{ ip})"$  by simp

thus " $\text{sqn } (\text{rt } \xi) \text{ ip} \leq \text{sqn } (\text{invalidate } (\text{rt } \xi) (\text{dests } \xi)) \text{ ip}"$

by (metis domI invalidate\_sqn option.sel)

qed

} note solve\_invalidate = this

show ?thesis

```
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]
onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn]
simp add: solve_invalidate)
```

qed

Proposition 7.7

lemma ip\_constant:

```
"paodv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_). ip \xi = i$ )"
by (inv_cterms simp add:  $\sigma_{AODV\_def}$ )
```

Proposition 7.8

lemma sender\_ip\_valid':

```
"paodv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_). \text{anycast } (\lambda m. \text{not\_Pkt } m \longrightarrow \text{msg\_sender } m = ip \xi) a$ )"
by inv_cterms
```

lemma sender\_ip\_valid:

```
"paodv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_). \text{anycast } (\lambda m. \text{not\_Pkt } m \longrightarrow \text{msg\_sender } m = i) a$ )"
by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid'])
```

(auto dest!: onlD onl1D)

lemma received\_msg\_inv:

"paadv i  $\models$  (recvm<sub>msg</sub> P  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l). l \in \{PAadv-:1\} \rightarrow P$  (msg  $\xi$ ))"  
by inv\_cterms

Proposition 7.9

lemma sip\_not\_ip':

"paadv i  $\models$  (recvm<sub>msg</sub> ( $\lambda m. not\_Pkt\ m \rightarrow msg\_sender\ m \neq i$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_). sip\ \xi \neq ip\ \xi$ )"  
by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]  
onl\_invariant\_sterms [OF aadv\_wf ip\_constant [THEN invariant\_restrict\_inD]]  
simp add: clear\_locals\_sip\_not\_ip') clarsimp+

lemma sip\_not\_ip:

"paadv i  $\models$  (recvm<sub>msg</sub> ( $\lambda m. not\_Pkt\ m \rightarrow msg\_sender\ m \neq i$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_). sip\ \xi \neq i$ )"  
by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]  
onl\_invariant\_sterms [OF aadv\_wf ip\_constant [THEN invariant\_restrict\_inD]]  
simp add: clear\_locals\_sip\_not\_ip') clarsimp+

Neither sip\_not\_ip' nor sip\_not\_ip is needed to show loop freedom.

Proposition 7.10

lemma hop\_count\_positive:

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_). \forall ip \in kD\ (rt\ \xi). the\ (dhops\ (rt\ \xi)\ ip) \geq 1$ )"  
by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf addpreRT\_welldefined]) auto

lemma rreq\_dip\_in\_vD\_dip\_eq\_ip:

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l). (l \in \{PRreq-:16..PRreq-:18\} \rightarrow dip\ \xi \in vD(rt\ \xi))$   
 $\wedge (l \in \{PRreq-:5, PRreq-:6\} \rightarrow dip\ \xi = ip\ \xi)$   
 $\wedge (l \in \{PRreq-:15..PRreq-:18\} \rightarrow dip\ \xi \neq ip\ \xi)$ )"

proof (inv\_cterms, elim conjE)

fix l  $\xi$  pp p'

assume "( $\xi, pp$ )  $\in$  reachable (paadv i) TT"

and "{PRreq-:17}  $\llbracket \lambda \xi. \xi \llbracket rt := the\ (addpreRT\ (rt\ \xi)\ (oip\ \xi)\ \{the\ (nhop\ (rt\ \xi)\ (dip\ \xi))\} \rrbracket \rrbracket p'$   
 $\in$  sterms  $\Gamma_{AODV}$  pp"

and "l = PRreq-:17"

and "dip  $\xi \in vD\ (rt\ \xi)$ "

from this(1-3) have "oip  $\xi \in kD\ (rt\ \xi)$ "

by (auto dest: onl\_invariant\_sterms [OF aadv\_wf addpreRT\_welldefined, where l="PRreq-:17"])

with  $\langle dip\ \xi \in vD\ (rt\ \xi) \rangle$

show "dip  $\xi \in vD\ (the\ (addpreRT\ (rt\ \xi)\ (oip\ \xi)\ \{the\ (nhop\ (rt\ \xi)\ (dip\ \xi))\})$ " by simp

qed

Proposition 7.11

lemma anycast\_msg\_zhops:

" $\bigwedge rreqid\ dip\ dsn\ dsk\ oip\ osn\ sip.$

paadv i  $\models_A$  onl1  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_). anycast\ msg\_zhops\ a$ )"

proof (inv\_cterms inv add:

onl\_invariant\_sterms [OF aadv\_wf rreq\_dip\_in\_vD\_dip\_eq\_ip [THEN invariant\_restrict\_inD]]

onl\_invariant\_sterms [OF aadv\_wf hop\_count\_positive [THEN invariant\_restrict\_inD]],

elim conjE)

fix l  $\xi$  a pp p' pp'

assume "( $\xi, pp$ )  $\in$  reachable (paadv i) TT"

and "{PRreq-:18}unicast( $\lambda \xi. the\ (nhop\ (rt\ \xi)\ (oip\ \xi)),$

$\lambda \xi. Rrep\ (the\ (dhops\ (rt\ \xi)\ (dip\ \xi)))\ (dip\ \xi)\ (sqn\ (rt\ \xi)\ (dip\ \xi))\ (oip\ \xi)\ (ip\ \xi).$

$p' \triangleright pp' \in$  sterms  $\Gamma_{AODV}$  pp"

and "l = PRreq-:18"

and "a = unicast (the (nhop (rt  $\xi$ ) (oip  $\xi$ )))

(Rrep (the (dhops (rt  $\xi$ ) (dip  $\xi$ ))) (dip  $\xi$ ) (sqn (rt  $\xi$ ) (dip  $\xi$ )) (oip  $\xi$ ) (ip  $\xi$ ))"

and \*: " $\forall ip \in kD\ (rt\ \xi). Suc\ 0 \leq the\ (dhops\ (rt\ \xi)\ ip)$ "

and "dip  $\xi \in vD\ (rt\ \xi)$ "

from  $\langle dip\ \xi \in vD\ (rt\ \xi) \rangle$  have "dip  $\xi \in kD\ (rt\ \xi)$ "

by (rule vD\_iD\_gives\_kD(1))

with \* have "Suc 0  $\leq$  the (dhops (rt  $\xi$ ) (dip  $\xi$ ))" ..

thus "0 < the (dhops (rt  $\xi$ ) (dip  $\xi$ ))" by simp  
qed

lemma hop\_count\_zero\_oip\_dip\_sip:

"paadv i  $\models$  (recvmsg msg\_zhops  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .  
 $(l \in \{PAadv-:4..PAadv-:5\} \cup \{PRreq-:n/n. True\} \rightarrow$   
 $(hops \xi = 0 \rightarrow oip \xi = sip \xi))$   
 $\wedge$   
 $((l \in \{PAadv-:6..PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow$   
 $(hops \xi = 0 \rightarrow dip \xi = sip \xi))))$ "  
by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]) auto

lemma osn\_rreq:

"paadv i  $\models$  (recvmsg rreq\_rrep\_sn  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .  
 $l \in \{PAadv-:4, PAadv-:5\} \cup \{PRreq-:n/n. True\} \rightarrow 1 \leq osn \xi$ )"  
by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]) clarsimp

lemma osn\_rreq':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq_rrep_sn m \wedge msg_zhops m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .  
 $l \in \{PAadv-:4, PAadv-:5\} \cup \{PRreq-:n/n. True\} \rightarrow 1 \leq osn \xi$ )"  
proof (rule invariant\_weakenE [OF osn\_rreq])

fix a  
assume "recvmsg ( $\lambda m. rreq_rrep_sn m \wedge msg_zhops m$ ) a"  
thus "recvmsg rreq\_rrep\_sn a"  
by (cases a) simp\_all  
qed

lemma dsn\_rrep:

"paadv i  $\models$  (recvmsg rreq\_rrep\_sn  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .  
 $l \in \{PAadv-:6, PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow 1 \leq dsn \xi$ )"  
by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]) clarsimp

lemma dsn\_rrep':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq_rrep_sn m \wedge msg_zhops m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .  
 $l \in \{PAadv-:6, PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow 1 \leq dsn \xi$ )"  
proof (rule invariant\_weakenE [OF dsn\_rrep])

fix a  
assume "recvmsg ( $\lambda m. rreq_rrep_sn m \wedge msg_zhops m$ ) a"  
thus "recvmsg rreq\_rrep\_sn a"  
by (cases a) simp\_all  
qed

lemma hop\_count\_zero\_oip\_dip\_sip':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq_rrep_sn m \wedge msg_zhops m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .  
 $(l \in \{PAadv-:4..PAadv-:5\} \cup \{PRreq-:n/n. True\} \rightarrow$   
 $(hops \xi = 0 \rightarrow oip \xi = sip \xi))$   
 $\wedge$   
 $((l \in \{PAadv-:6..PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow$   
 $(hops \xi = 0 \rightarrow dip \xi = sip \xi))))$ "

proof (rule invariant\_weakenE [OF hop\_count\_zero\_oip\_dip\_sip])  
fix a  
assume "recvmsg ( $\lambda m. rreq_rrep_sn m \wedge msg_zhops m$ ) a"  
thus "recvmsg msg\_zhops a"  
by (cases a) simp\_all  
qed

Proposition 7.12

lemma zero\_seq\_unk\_hops\_one':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq_rrep_sn m \wedge msg_zhops m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_)$ .  
 $\forall dip \in kD(rt \xi). (sqn (rt \xi) dip = 0 \rightarrow sqnf (rt \xi) dip = unk)$   
 $\wedge (sqnf (rt \xi) dip = unk \rightarrow the (dhops (rt \xi) dip) = 1)$   
 $\wedge (the (dhops (rt \xi) dip) = 1 \rightarrow the (nhop (rt \xi) dip) = dip))$ "

proof -

{ fix dip and  $\xi :: state$  and  $P$

```

assume "sqn (invalidate (rt ξ) (dests ξ)) dip = 0"
  and all: "∀ ip. sqn (rt ξ) ip ≤ sqn (invalidate (rt ξ) (dests ξ)) ip"
  and *: "sqn (rt ξ) dip = 0 ⇒ P ξ dip"
have "P ξ dip"
proof -
  from all have "sqn (rt ξ) dip ≤ sqn (invalidate (rt ξ) (dests ξ)) dip" ..
  with ⟨sqn (invalidate (rt ξ) (dests ξ)) dip = 0⟩ have "sqn (rt ξ) dip = 0" by simp
  thus "P ξ dip" by (rule *)
qed
} note sqn_invalidate_zero [elim!] = this

{ fix dsn hops :: nat and sip oip rt and ip dip :: ip
  assume "∀ dip ∈ kD(rt).
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
    and "hops = 0 → sip = dip"
    and "Suc 0 ≤ dsn"
    and "ip ≠ dip → ip ∈ kD(rt)"
  hence "the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, { }))) ip = Suc 0 →
    the (nhop (update rt dip (dsn, kno, val, Suc hops, sip, { }))) ip = ip"
    by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
} note prreq_ok1 [simp] = this

{ fix ip dsn hops sip oip rt dip
  assume "∀ dip ∈ kD(rt).
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
    and "Suc 0 ≤ dsn"
    and "ip ≠ dip → ip ∈ kD(rt)"
  hence "π3(the (update rt dip (dsn, kno, val, Suc hops, sip, { }))) ip = unk →
    the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, { }))) ip = Suc 0"
    by - (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)
} note prreq_ok2 [simp] = this

{ fix ip dsn hops sip oip rt dip
  assume "∀ dip ∈ kD(rt).
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
    and "Suc 0 ≤ dsn"
    and "ip ≠ dip → ip ∈ kD(rt)"
  hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip, { }))) ip = 0 →
    π3 (the (update rt dip (dsn, kno, val, Suc hops, sip, { }))) ip = unk"
    by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
} note prreq_ok3 [simp] = this

{ fix rt sip
  assume "∀ dip ∈ kD rt.
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
  hence "∀ dip ∈ kD rt.
    (sqn (update rt sip (0, unk, val, Suc 0, sip, { }))) dip = 0 →
    π3(the (update rt sip (0, unk, val, Suc 0, sip, { }))) dip = unk)
    ∧ (π3(the (update rt sip (0, unk, val, Suc 0, sip, { }))) dip = unk →
    the (dhops (update rt sip (0, unk, val, Suc 0, sip, { }))) dip = Suc 0)
    ∧ (the (dhops (update rt sip (0, unk, val, Suc 0, sip, { }))) dip = Suc 0 →
    the (nhop (update rt sip (0, unk, val, Suc 0, sip, { }))) dip = dip)"
    by - (rule update_cases, simp_all add: sqnf_def sqn_def)
} note prreq_ok4 [simp] = this

have prreq_ok5 [simp]: "∧ sip rt.

```

```

 $\pi_3(\text{the } (\text{update } \text{rt } \text{sip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{\}) \text{ sip})) = \text{unk} \longrightarrow$ 
 $\text{the } (\text{dhops } (\text{update } \text{rt } \text{sip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{\}) \text{ sip})) \text{ sip} = \text{Suc } 0"$ 
by (rule update_cases) simp_all

have prreq_ok6 [simp]: " $\wedge \text{sip } \text{rt}.$ 
sqn (update rt sip (0, unk, val, Suc 0, sip, {})) sip = 0  $\longrightarrow$ 
 $\pi_3(\text{the } (\text{update } \text{rt } \text{sip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{\}) \text{ sip})) = \text{unk}"$ 
by (rule update_cases) simp_all

show ?thesis
by (inv_terms inv add: onl_invariant_sterms_TT [OF aadv_wf addpreRT_welldefined]
onl_invariant_sterms [OF aadv_wf hop_count_zero_oip_dip_sip']
seq_step_invariant_sterms_TT [OF sqns_increase aadv_wf aadv_trans]
onl_invariant_sterms [OF aadv_wf osn_rreq']
onl_invariant_sterms [OF aadv_wf dsn_rrep']) clarsimp+

qed

lemma zero_seq_unk_hops_one:
"paadv i  $\models$  (recvmmsg ( $\lambda m. \text{rreq\_rrep\_sn } m \wedge \text{msg\_zhops } m$ )  $\longrightarrow$ ) onl  $\Gamma_{AODV} (\lambda(\xi, \_).$ 
 $\forall \text{dip} \in \text{kD}(\text{rt } \xi). (\text{sqn } (\text{rt } \xi) \text{ dip} = 0 \longrightarrow (\text{sqnf } (\text{rt } \xi) \text{ dip} = \text{unk}$ 
 $\wedge \text{the } (\text{dhops } (\text{rt } \xi) \text{ dip}) = 1$ 
 $\wedge \text{the } (\text{nhop } (\text{rt } \xi) \text{ dip}) = \text{dip})))"$ 
by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

lemma kD_unk_or_atleast_one:
"paadv i  $\models$  (recvmmsg rreq_rrep_sn  $\longrightarrow$ ) onl  $\Gamma_{AODV} (\lambda(\xi, 1).$ 
 $\forall \text{dip} \in \text{kD}(\text{rt } \xi). \pi_3(\text{the } (\text{rt } \xi \text{ dip})) = \text{unk} \vee 1 \leq \pi_2(\text{the } (\text{rt } \xi \text{ dip})))"$ 

proof -
{ fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
assume "dsk1 = unk  $\vee$  Suc 0  $\leq$  dsn2"
hence " $\pi_3(\text{the } (\text{update } \text{rt } \text{sip } (\text{dsn1}, \text{dsk1}, \text{flag1}, \text{hops1}, \text{nhip1}, \text{pre1}) \text{ sip})) = \text{unk}$ 
 $\vee$  Suc 0  $\leq$  sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2)) sip"
unfolding update_def by (cases "dsk1 =unk") (clarsimp split: option.split)+
} note fromsip [simp] = this

{ fix dip sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
assume allkd: " $\forall \text{dip} \in \text{kD}(\text{rt}). \pi_3(\text{the } (\text{rt } \text{dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } \text{rt } \text{dip}"$ 
and **: "dsk1 = unk  $\vee$  Suc 0  $\leq$  dsn2"
have " $\forall \text{dip} \in \text{kD}(\text{rt}). \pi_3(\text{the } (\text{update } \text{rt } \text{sip } (\text{dsn1}, \text{dsk1}, \text{flag1}, \text{hops1}, \text{nhip1}, \text{pre1}) \text{ dip})) = \text{unk}$ 
 $\vee$  Suc 0  $\leq$  sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2)) dip"
(is " $\forall \text{dip} \in \text{kD}(\text{rt}). ?\text{prop } \text{dip}"$ )
proof
fix dip
assume "dip  $\in$  kD(rt)"
thus "?prop dip"
proof (cases "dip = sip")
assume "dip = sip"
with ** show ?thesis
by simp
next
assume "dip  $\neq$  sip"
with (dip  $\in$  kD(rt)) allkd show ?thesis
by simp
qed
qed
} note solve_update [simp] = this

{ fix dip rt dests
assume *: " $\forall \text{ip} \in \text{dom}(\text{dests}). \text{ip} \in \text{kD}(\text{rt}) \wedge \text{sqn } \text{rt } \text{ip} \leq \text{the } (\text{dests } \text{ip})"$ 
and **: " $\forall \text{ip} \in \text{kD}(\text{rt}). \pi_3(\text{the } (\text{rt } \text{ip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } \text{rt } \text{ip}"$ 
have " $\forall \text{dip} \in \text{kD}(\text{rt}). \pi_3(\text{the } (\text{rt } \text{dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } (\text{invalidate } \text{rt } \text{dests}) \text{ dip}"$ 
proof
fix dip
assume "dip  $\in$  kD(rt)"

```



```

with ** have "π3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip" ..
thus "π3 (the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"
proof
  assume "π3(the (rt dip)) = unk" thus ?thesis ..
next
  assume "Suc 0 ≤ sqn rt dip"
  have "Suc 0 ≤ sqn (invalidate rt dests) dip"
  proof (cases "dip ∈ dom(dests)")
    assume "dip ∈ dom(dests)"
    with * have "sqn rt dip ≤ the (dests dip)" by simp
    with ⟨Suc 0 ≤ sqn rt dip⟩ have "Suc 0 ≤ the (dests dip)" by simp
    with ⟨dip ∈ dom(dests)⟩ ⟨dip ∈ kD(rt)⟩ [THEN kD_Some] show ?thesis
      unfolding invalidate_def sqn_def by auto
  next
    assume "dip ∉ dom(dests)"
    with ⟨Suc 0 ≤ sqn rt dip⟩ ⟨dip ∈ kD(rt)⟩ [THEN kD_Some] show ?thesis
      unfolding invalidate_def sqn_def by auto
  qed
thus ?thesis by (rule disjI2)
qed
qed
} note solve_invalidate [simp] = this

show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms_TT [OF aadv_wf addpreRT_welldefined]
      onl_invariant_sterms [OF aadv_wf dests_bigger_than_sqn
                            [THEN invariant_restrict_inD]]
      onl_invariant_sterms [OF aadv_wf osn_rreq]
      onl_invariant_sterms [OF aadv_wf dsn_rrep]
      simp add: proj3_inv proj2_eq_sqn)

```

qed

Proposition 7.13

lemma rreq\_rrep\_sn\_any\_step\_invariant:

"paadv i ⊨<sub>A</sub> (recvmmsg rreq\_rrep\_sn →) onll Γ<sub>AODV</sub> (λ(., a, \_). anycast rreq\_rrep\_sn a)"

proof -

have sqnf\_kno: "paadv i ⊨ onl Γ<sub>AODV</sub> (λ(ξ, l).

(l ∈ {PRreq-:16..PRreq-:18} → sqnf (rt ξ) (dip ξ) = kno))"

by (inv\_cterms inv add: onl\_invariant\_sterms\_TT [OF aadv\_wf addpreRT\_welldefined])

show ?thesis

by (inv\_cterms inv add: onl\_invariant\_sterms\_TT [OF aadv\_wf addpreRT\_welldefined]
 onl\_invariant\_sterms [OF aadv\_wf sequence\_number\_one\_or\_bigger
 [THEN invariant\_restrict\_inD]]
 onl\_invariant\_sterms [OF aadv\_wf kD\_unk\_or\_atleast\_one]
 onl\_invariant\_sterms\_TT [OF aadv\_wf sqnf\_kno]
 onl\_invariant\_sterms [OF aadv\_wf osn\_rreq]
 onl\_invariant\_sterms [OF aadv\_wf dsn\_rrep])

(auto simp: proj2\_eq\_sqn)

qed

Proposition 7.14

lemma rreq\_rrep\_fresh\_any\_step\_invariant:

"paadv i ⊨<sub>A</sub> onll Γ<sub>AODV</sub> (λ((ξ, \_), a, \_). anycast (rreq\_rrep\_fresh (rt ξ)) a)"

proof -

have rreq\_oip: "paadv i ⊨ onl Γ<sub>AODV</sub> (λ(ξ, l).

(l ∈ {PRreq-:3, PRreq-:4, PRreq-:15, PRreq-:27}

→ oip ξ ∈ kD(rt ξ)

∧ (sqn (rt ξ) (oip ξ) > (osn ξ)

∨ (sqn (rt ξ) (oip ξ) = (osn ξ)

∧ the (dhops (rt ξ) (oip ξ)) ≤ Suc (hops ξ)

∧ the (flag (rt ξ) (oip ξ)) = val))))"

proof inv\_cterms

fix l ξ l' pp p'

assume "(ξ, pp) ∈ reachable (paadv i) TT"

```

and "{PRreq-:2}[[λξ. ξ(rt :=
  update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})]]] p' ∈ sterms ΓAODV pp"
and "l' = PRreq-:3"
show "osn ξ < sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})) (oip ξ)
  ∨ (sqn (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})) (oip ξ) = osn ξ
    ∧ the (dhops (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})) (oip ξ))
      ≤ Suc (hops ξ)
    ∧ the (flag (update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})) (oip ξ))
      = val)"
unfolding update_def by (clarsimp split: option.split)
  (metis linorder_neqE_nat not_less)
qed

```

```

have rrep_prrep: "paadv i ⊨ onl ΓAODV (λ(ξ, l).
  (l ∈ {PRrep-:2..PRrep-:7} → (dip ξ ∈ kD(rt ξ)
    ∧ sqn (rt ξ) (dip ξ) = dsn ξ
    ∧ the (dhops (rt ξ) (dip ξ)) = Suc (hops ξ)
    ∧ the (flag (rt ξ) (dip ξ)) = val
    ∧ the (nhop (rt ξ) (dip ξ)) ∈ kD (rt ξ))))"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf rrep_1_update_changes]
  onl_invariant_sterms [OF aadv_wf sip_in_kD])

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf rreq_oip]
  onl_invariant_sterms [OF aadv_wf rreq_dip_in_vD_dip_eq_ip]
  onl_invariant_sterms [OF aadv_wf rrep_prrep])

```

qed

Proposition 7.15

lemma rerr\_invalid\_any\_step\_invariant:

"paadv i ⊨<sub>A</sub> onll Γ<sub>AODV</sub> (λ((ξ, \_), a, \_). anycast (rerr\_invalid (rt ξ)) a)"

proof -

```

have dests_inv: "paadv i ⊨
  onl ΓAODV (λ(ξ, l). (l ∈ {PAadv-:15, PPkt-:7, PRreq-:9,
    PRreq-:21, PRrep-:10, PRerr-:1}
    → (∀ ip ∈ dom(dests ξ). ip ∈ vD(rt ξ)))
  ∧ (l ∈ {PAadv-:16..PAadv-:19}
    ∪ {PPkt-:8..PPkt-:11}
    ∪ {PRreq-:10..PRreq-:13}
    ∪ {PRreq-:22..PRreq-:25}
    ∪ {PRrep-:11..PRrep-:14}
    ∪ {PRerr-:2..PRerr-:5} → (∀ ip ∈ dom(dests ξ). ip ∈ iD(rt ξ)
    ∧ the (dests ξ ip) = sqn (rt ξ) ip))
  ∧ (l = PPkt-:14 → dip ξ ∈ iD(rt ξ)))"

```

by inv\_cterms (clarsimp split: split\_if\_asm option.split\_asm simp: domIff)+

show ?thesis

by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf dests\_inv])

qed

Proposition 7.16

Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs  $(rip, rsn)$  is assigned to the variable  $dests$  of type  $ip \rightarrow sqn$ , or to the first argument of the function  $rerr$ , this set is a partial function, i.e., there is at most one entry  $(rip, rsn)$  for each destination  $rip$ : guaranteed by type.

lemma dests\_vD\_inc\_sqn:

"paadv i ⊨

```

onl  $\Gamma_{AODV} (\lambda(\xi, l). (l \in \{PAadv-:15, PPkt-:7, PRreq-:9, PRreq-:21, PRrep-:10\}
\rightarrow (\forall ip \in \text{dom}(\text{dests } \xi). ip \in vD(\text{rt } \xi) \wedge \text{the } (\text{dests } \xi \text{ } ip) = \text{inc } (\text{sqn } (\text{rt } \xi) \text{ } ip)))
\wedge (l = PRerr-:1
\rightarrow (\forall ip \in \text{dom}(\text{dests } \xi). ip \in vD(\text{rt } \xi) \wedge \text{the } (\text{dests } \xi \text{ } ip) > \text{sqn } (\text{rt } \xi) \text{ } ip)))"$ 
by inv_cterms (clarsimp split: split_if_asm option.split_asm)+

```

Proposition 7.27

lemma route\_tables\_fresher:

```

"paadv i  $\models_A$  (recvmgs rreq_rrep_sn  $\rightarrow$ ) onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)).
\forall dip \in kD(\text{rt } \xi). \text{rt } \xi \sqsubseteq_{dip} \text{rt } \xi')$ "

```

proof (inv\_cterms inv add:

```

onl_invariant_sterms [OF aadv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aadv_wf hop_count_positive [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aadv_wf osn_rreq]
onl_invariant_sterms [OF aadv_wf dsn_rrep]
onl_invariant_sterms [OF aadv_wf addpreRT_welldefined [THEN invariant_restrict_inD]]

```

fix  $\xi$  pp p'

assume " $(\xi, pp) \in \text{reachable } (\text{paadv } i) (\text{recvmgs } rreq\_rrep\_sn)$ "

and " $\{PRreq-:2\} \llbracket \lambda \xi. \xi (\text{rt} := \text{update } (\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc } (\text{hops } \xi), \text{sip } \xi, \{ }) \rrbracket$   
 $p' \in \text{sterms } \Gamma_{AODV} \text{ } pp$ "

and " $\text{Suc } 0 \leq \text{osn } \xi$ "

and \*: " $\forall ip \in kD(\text{rt } \xi). \text{Suc } 0 \leq \text{the } (\text{dhops } (\text{rt } \xi) \text{ } ip)$ "

show " $\forall ip \in kD(\text{rt } \xi). \text{rt } \xi \sqsubseteq_{ip} \text{update } (\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc } (\text{hops } \xi), \text{sip } \xi, \{ })$ "

proof

fix ip

assume " $ip \in kD(\text{rt } \xi)$ "

moreover with \* have " $1 \leq \text{the } (\text{dhops } (\text{rt } \xi) \text{ } ip)$ " by simp

moreover from  $\langle \text{Suc } 0 \leq \text{osn } \xi \rangle$

have " $\text{update\_arg\_wf } (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc } (\text{hops } \xi), \text{sip } \xi, \{ })$ " ..

ultimately show " $\text{rt } \xi \sqsubseteq_{ip} \text{update } (\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc } (\text{hops } \xi), \text{sip } \xi, \{ })$ "

by (rule rt\_fresher\_update)

qed

next

fix  $\xi$  pp p'

assume " $(\xi, pp) \in \text{reachable } (\text{paadv } i) (\text{recvmgs } rreq\_rrep\_sn)$ "

and " $\{PRrep-:1\} \llbracket \lambda \xi. \xi (\text{rt} := \text{update } (\text{rt } \xi) (\text{dip } \xi) (\text{dsn } \xi, \text{kno}, \text{val}, \text{Suc } (\text{hops } \xi), \text{sip } \xi, \{ }) \rrbracket$   
 $p' \in \text{sterms } \Gamma_{AODV} \text{ } pp$ "

and " $\text{Suc } 0 \leq \text{dsn } \xi$ "

and \*: " $\forall ip \in kD(\text{rt } \xi). \text{Suc } 0 \leq \text{the } (\text{dhops } (\text{rt } \xi) \text{ } ip)$ "

show " $\forall ip \in kD(\text{rt } \xi). \text{rt } \xi \sqsubseteq_{ip} \text{update } (\text{rt } \xi) (\text{dip } \xi) (\text{dsn } \xi, \text{kno}, \text{val}, \text{Suc } (\text{hops } \xi), \text{sip } \xi, \{ })$ "

proof

fix ip

assume " $ip \in kD(\text{rt } \xi)$ "

moreover with \* have " $1 \leq \text{the } (\text{dhops } (\text{rt } \xi) \text{ } ip)$ " by simp

moreover from  $\langle \text{Suc } 0 \leq \text{dsn } \xi \rangle$

have " $\text{update\_arg\_wf } (\text{dsn } \xi, \text{kno}, \text{val}, \text{Suc } (\text{hops } \xi), \text{sip } \xi, \{ })$ " ..

ultimately show " $\text{rt } \xi \sqsubseteq_{ip} \text{update } (\text{rt } \xi) (\text{dip } \xi) (\text{dsn } \xi, \text{kno}, \text{val}, \text{Suc } (\text{hops } \xi), \text{sip } \xi, \{ })$ "

by (rule rt\_fresher\_update)

qed

qed

end

## 1.7 The quality increases predicate

theory A\_Quality\_Increases

imports A\_Aadv\_Predicates A\_Fresher

begin

definition quality\_increases :: "state  $\Rightarrow$  state  $\Rightarrow$  bool"

where "quality\_increases  $\xi$   $\xi'$   $\equiv$   $(\forall dip \in kD(\text{rt } \xi). \text{dip} \in kD(\text{rt } \xi') \wedge \text{rt } \xi \sqsubseteq_{dip} \text{rt } \xi') \wedge (\forall dip. \text{sqn } (\text{rt } \xi) \text{ } dip \leq \text{sqn } (\text{rt } \xi') \text{ } dip)$ "

lemma quality\_increasesI [intro!]:

```

assumes "\^dip. dip ∈ kD(rt ξ) ⇒ dip ∈ kD(rt ξ')"
  and "\^dip. [ dip ∈ kD(rt ξ); dip ∈ kD(rt ξ') ] ⇒ rt ξ ⊑dip rt ξ'"
  and "\^dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip"
shows "quality_increases ξ ξ'"
unfolding quality_increases_def using assms by clarsimp

lemma quality_increasesE [elim]:
  fixes dip
  assumes "quality_increases ξ ξ'"
    and "dip ∈ kD(rt ξ)"
    and "[ dip ∈ kD(rt ξ'); rt ξ ⊑dip rt ξ'; sqn (rt ξ) dip ≤ sqn (rt ξ') dip ] ⇒ R dip ξ ξ'"
  shows "R dip ξ ξ'"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_rt_fresherD [dest]:
  fixes ip
  assumes "quality_increases ξ ξ'"
    and "ip ∈ kD(rt ξ)"
  shows "rt ξ ⊑ip rt ξ'"
using assms by auto

lemma quality_increases_sqnE [elim]:
  fixes dip
  assumes "quality_increases ξ ξ'"
    and "sqn (rt ξ) dip ≤ sqn (rt ξ') dip ⇒ R dip ξ ξ'"
  shows "R dip ξ ξ'"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_refl [intro, simp]: "quality_increases ξ ξ"
by rule simp_all

lemma strictly_fresher_quality_increases_right [elim]:
  fixes σ σ' dip
  assumes "rt (σ i) ⊑dip rt (σ nhip)"
    and qinc: "quality_increases (σ nhip) (σ' nhip)"
    and "dip ∈ kD(rt (σ nhip))"
  shows "rt (σ i) ⊑dip rt (σ' nhip)"
proof -
  from qinc have "rt (σ nhip) ⊑dip rt (σ' nhip)" using ⟨dip ∈ kD(rt (σ nhip))⟩
  by auto
  with ⟨rt (σ i) ⊑dip rt (σ nhip)⟩ show ?thesis ..
qed

lemma kD_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
  shows "i ∈ kD(rt ξ')"
using assms by auto

lemma kD_nsqn_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
  shows "i ∈ kD(rt ξ') ∧ nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
proof -
  from assms have "i ∈ kD(rt ξ')" ..
  moreover with assms have "rt ξ ⊑i rt ξ'" by auto
  ultimately have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
  using ⟨i ∈ kD(rt ξ)⟩ by - (erule(2) rt_fresher_imp_nsqn_le)
  with ⟨i ∈ kD(rt ξ')⟩ show ?thesis ..
qed

lemma nsqn_quality_increases [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"

```

```

  shows "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi'$ ) i"
using assms by (rule kD_nsqn_quality_increases [THEN conjunct2])

lemma kD_nsqn_quality_increases_trans [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "s  $\leq$  nsqn (rt  $\xi$ ) i"
    and "quality_increases  $\xi$   $\xi'$ "
  shows "i $\in$ kD(rt  $\xi'$ )  $\wedge$  s  $\leq$  nsqn (rt  $\xi'$ ) i"
proof
  from (i $\in$ kD(rt  $\xi$ )) and (quality_increases  $\xi$   $\xi'$ ) show "i $\in$ kD(rt  $\xi'$ )" ..
next
  from (i $\in$ kD(rt  $\xi$ )) and (quality_increases  $\xi$   $\xi'$ ) have "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi'$ ) i" ..
  with (s  $\leq$  nsqn (rt  $\xi$ ) i) show "s  $\leq$  nsqn (rt  $\xi'$ ) i" by (rule le_trans)
qed

lemma nsqn_quality_increases_nsqn_lt_lt [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "quality_increases  $\xi$   $\xi'$ "
    and "s < nsqn (rt  $\xi$ ) i"
  shows "s < nsqn (rt  $\xi'$ ) i"
proof -
  from assms(1-2) have "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi'$ ) i" ..
  with (s < nsqn (rt  $\xi$ ) i) show "s < nsqn (rt  $\xi'$ ) i" by simp
qed

lemma nsqn_quality_increases_dhops [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "quality_increases  $\xi$   $\xi'$ "
    and "nsqn (rt  $\xi$ ) i = nsqn (rt  $\xi'$ ) i"
  shows "the (dhops (rt  $\xi$ ) i)  $\geq$  the (dhops (rt  $\xi'$ ) i)"
using assms unfolding quality_increases_def
by (clarsimp) (drule(1) bspec, clarsimp simp: rt_fresher_def2)

lemma nsqn_quality_increases_nsqn_eq_le [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "quality_increases  $\xi$   $\xi'$ "
    and "s = nsqn (rt  $\xi$ ) i"
  shows "s < nsqn (rt  $\xi'$ ) i  $\vee$  (s = nsqn (rt  $\xi'$ ) i  $\wedge$  the (dhops (rt  $\xi$ ) i)  $\geq$  the (dhops (rt  $\xi'$ ) i))"
using assms by (metis nat_less_le nsqn_quality_increases nsqn_quality_increases_dhops)

lemma quality_increases_rreq_rrep_props [elim]:
  fixes sn ip hops sip
  assumes qinc: "quality_increases ( $\sigma$  sip) ( $\sigma'$  sip)"
    and "1  $\leq$  sn"
    and *: "ip $\in$ kD(rt ( $\sigma$  sip))  $\wedge$  sn  $\leq$  nsqn (rt ( $\sigma$  sip)) ip
       $\wedge$  (nsqn (rt ( $\sigma$  sip)) ip = sn
         $\longrightarrow$  (the (dhops (rt ( $\sigma$  sip)) ip)  $\leq$  hops
           $\vee$  the (flag (rt ( $\sigma$  sip)) ip) = inv))"
  shows "ip $\in$ kD(rt ( $\sigma'$  sip))  $\wedge$  sn  $\leq$  nsqn (rt ( $\sigma'$  sip)) ip
     $\wedge$  (nsqn (rt ( $\sigma'$  sip)) ip = sn
       $\longrightarrow$  (the (dhops (rt ( $\sigma'$  sip)) ip)  $\leq$  hops
         $\vee$  the (flag (rt ( $\sigma'$  sip)) ip) = inv))"
    (is "_  $\wedge$  ?nsqnafter")
proof -
  from * obtain "ip $\in$ kD(rt ( $\sigma$  sip))" and "sn  $\leq$  nsqn (rt ( $\sigma$  sip)) ip" by auto

  from (quality_increases ( $\sigma$  sip) ( $\sigma'$  sip))
    have "sqn (rt ( $\sigma$  sip)) ip  $\leq$  sqn (rt ( $\sigma'$  sip)) ip" ..
  from (quality_increases ( $\sigma$  sip) ( $\sigma'$  sip)) and (ip $\in$ kD (rt ( $\sigma$  sip)))
    have "ip $\in$ kD (rt ( $\sigma'$  sip))" ..

  from (sn  $\leq$  nsqn (rt ( $\sigma$  sip)) ip) have ?nsqnafter
proof
  assume "sn < nsqn (rt ( $\sigma$  sip)) ip"

```

```

also from  $\langle ip \in kD(rt(\sigma sip)) \rangle$  and  $\langle quality\_increases(\sigma sip)(\sigma' sip) \rangle$ 
  have "...  $\leq nsqn(rt(\sigma' sip)) ip$ " ..
finally have " $sn < nsqn(rt(\sigma' sip)) ip$ " .
thus ?thesis by simp
next
assume " $sn = nsqn(rt(\sigma sip)) ip$ "
with  $\langle ip \in kD(rt(\sigma sip)) \rangle$  and  $\langle quality\_increases(\sigma sip)(\sigma' sip) \rangle$ 
  have " $sn < nsqn(rt(\sigma' sip)) ip$ "
     $\vee (sn = nsqn(rt(\sigma' sip)) ip$ 
       $\wedge the(dhops(rt(\sigma' sip)) ip) \leq the(dhops(rt(\sigma sip)) ip))$ " ..
hence " $sn < nsqn(rt(\sigma' sip)) ip$ "
     $\vee (nsqn(rt(\sigma' sip)) ip = sn \wedge (the(dhops(rt(\sigma' sip)) ip) \leq hops$ 
       $\vee the(flag(rt(\sigma' sip)) ip) = inv))$ "
proof
  assume " $sn < nsqn(rt(\sigma' sip)) ip$ " thus ?thesis ..
next
assume " $sn = nsqn(rt(\sigma' sip)) ip$ "
   $\wedge the(dhops(rt(\sigma sip)) ip) \geq the(dhops(rt(\sigma' sip)) ip)$ "
hence " $sn = nsqn(rt(\sigma' sip)) ip$ "
  and " $the(dhops(rt(\sigma' sip)) ip) \leq the(dhops(rt(\sigma sip)) ip)$ " by auto

from * and  $\langle sn = nsqn(rt(\sigma sip)) ip \rangle$  have " $the(dhops(rt(\sigma sip)) ip) \leq hops$ "
   $\vee the(flag(rt(\sigma sip)) ip) = inv$ "

  by simp
thus ?thesis
proof
  assume " $the(dhops(rt(\sigma sip)) ip) \leq hops$ "
  with  $\langle the(dhops(rt(\sigma' sip)) ip) \leq the(dhops(rt(\sigma sip)) ip) \rangle$ 
  have " $the(dhops(rt(\sigma' sip)) ip) \leq hops$ " by simp
  with  $\langle sn = nsqn(rt(\sigma' sip)) ip \rangle$  show ?thesis by simp
next
assume " $the(flag(rt(\sigma sip)) ip) = inv$ "
with  $\langle ip \in kD(rt(\sigma sip)) \rangle$  have " $nsqn(rt(\sigma sip)) ip = sqn(rt(\sigma sip)) ip - 1$ " ..

with  $\langle sn \geq 1 \rangle$  and  $\langle sn = nsqn(rt(\sigma sip)) ip \rangle$ 
  have " $sqn(rt(\sigma sip)) ip > 1$ " by simp

from  $\langle ip \in kD(rt(\sigma' sip)) \rangle$  show ?thesis
proof (rule vD_or_iD)
  assume " $ip \in iD(rt(\sigma' sip))$ "
  hence " $the(flag(rt(\sigma' sip)) ip) = inv$ " ..
  with  $\langle sn = nsqn(rt(\sigma' sip)) ip \rangle$  show ?thesis
  by simp
next
assume " $ip \in vD(rt(\sigma' sip))$ "
hence " $nsqn(rt(\sigma' sip)) ip = sqn(rt(\sigma' sip)) ip$ " ..
with  $\langle sqn(rt(\sigma sip)) ip \leq sqn(rt(\sigma' sip)) ip \rangle$ 
  have " $nsqn(rt(\sigma' sip)) ip \geq sqn(rt(\sigma sip)) ip$ " by simp

with  $\langle sqn(rt(\sigma sip)) ip > 1 \rangle$ 
  have " $nsqn(rt(\sigma' sip)) ip > sqn(rt(\sigma sip)) ip - 1$ " by simp
with  $\langle nsqn(rt(\sigma sip)) ip = sqn(rt(\sigma sip)) ip - 1 \rangle$ 
  have " $nsqn(rt(\sigma' sip)) ip > nsqn(rt(\sigma sip)) ip$ " by simp
with  $\langle sn = nsqn(rt(\sigma sip)) ip \rangle$  have " $nsqn(rt(\sigma' sip)) ip > sn$ "
  by simp
thus ?thesis ..
qed
qed
qed
thus ?thesis by (metis (mono_tags) le_cases not_le)
qed
with  $\langle ip \in kD(rt(\sigma' sip)) \rangle$  show " $ip \in kD(rt(\sigma' sip)) \wedge ?nsqnafter$ " ..
qed

```

lemma quality\_increases\_rreq\_rrep\_props':

fixes sn ip hops sip  
 assumes "∀j. quality\_increases (σ j) (σ' j)"  
 and "1 ≤ sn"  
 and \*: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip  
 ∧ (nsqn (rt (σ sip)) ip = sn  
 → (the (dhops (rt (σ sip)) ip) ≤ hops  
 ∨ the (flag (rt (σ sip)) ip) = inv))"  
 shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip  
 ∧ (nsqn (rt (σ' sip)) ip = sn  
 → (the (dhops (rt (σ' sip)) ip) ≤ hops  
 ∨ the (flag (rt (σ' sip)) ip) = inv))"

proof -

from assms(1) have "quality\_increases (σ sip) (σ' sip)" ..  
 thus ?thesis using assms(2-3) by (rule quality\_increases\_rreq\_rrep\_props)  
 qed

lemma rteq\_quality\_increases:

assumes "∀j. j ≠ i → quality\_increases (σ j) (σ' j)"  
 and "rt (σ' i) = rt (σ i)"  
 shows "∀j. quality\_increases (σ j) (σ' j)"  
 using assms by clarsimp (metis order\_refl quality\_increasesI rt\_fresher\_refl)

definition msg\_fresh :: "(ip ⇒ state) ⇒ msg ⇒ bool"

where "msg\_fresh σ m ≡

case m of Rreq hops c \_ \_ oipc osnc sipc ⇒ osnc ≥ 1 ∧ (sipc ≠ oipc →  
 oipc ∈ kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) oipc ≥ osnc  
 ∧ (nsqn (rt (σ sipc)) oipc = osnc  
 → (hops c ≥ the (dhops (rt (σ sipc)) oipc)  
 ∨ the (flag (rt (σ sipc)) oipc) = inv)))  
 | Rrep hops c dipc dsnc \_ sipc ⇒ dsnc ≥ 1 ∧ (sipc ≠ dipc →  
 dipc ∈ kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) dipc ≥ dsnc  
 ∧ (nsqn (rt (σ sipc)) dipc = dsnc  
 → (hops c ≥ the (dhops (rt (σ sipc)) dipc)  
 ∨ the (flag (rt (σ sipc)) dipc) = inv)))  
 | Rerr destsc sipc ⇒ (∀ripc ∈ dom(destsc). (ripc ∈ kD(rt (σ sipc))  
 ∧ the (destsc ripc) - 1 ≤ nsqn (rt (σ sipc)) ripc))  
 | \_ ⇒ True"

lemma msg\_fresh [simp]:

"∧hops dip dsn dsk oip osn sip.  
 msg\_fresh σ (Rreq hops dip dsn dsk oip osn sip) =  
 (osn ≥ 1 ∧ (sip ≠ oip → oip ∈ kD(rt (σ sip))  
 ∧ nsqn (rt (σ sip)) oip ≥ osn  
 ∧ (nsqn (rt (σ sip)) oip = osn  
 → (hops ≥ the (dhops (rt (σ sip)) oip)  
 ∨ the (flag (rt (σ sip)) oip) = inv)))))"  
 "∧hops dip dsn oip sip. msg\_fresh σ (Rrep hops dip dsn oip sip) =  
 (dsn ≥ 1 ∧ (sip ≠ dip → dip ∈ kD(rt (σ sip))  
 ∧ nsqn (rt (σ sip)) dip ≥ dsn  
 ∧ (nsqn (rt (σ sip)) dip = dsn  
 → (hops ≥ the (dhops (rt (σ sip)) dip)  
 ∨ the (flag (rt (σ sip)) dip) = inv)))))"  
 "∧dests sip. msg\_fresh σ (Rerr dests sip) =  
 (∀ripc ∈ dom(dests). (ripc ∈ kD(rt (σ sip))  
 ∧ the (dests ripc) - 1 ≤ nsqn (rt (σ sip)) ripc))"  
 "∧d dip. msg\_fresh σ (Newpkt d dip) = True"  
 "∧dip sip. msg\_fresh σ (Pkt d dip sip) = True"  
 unfolding msg\_fresh\_def by simp\_all

lemma msg\_fresh\_inc\_sn [simp, elim]:

"msg\_fresh σ m ⇒ rreq\_rrep\_sn m"  
 by (cases m) simp\_all

```

lemma recv_msg_fresh_inc_sn [simp, elim]:
  "orecvmsg (msg_fresh)  $\sigma$  m  $\implies$  recvmsg rreq_rrep_sn m"
  by (cases m) simp_all

lemma rreq_nsqn_is_fresh [simp]:
  fixes  $\sigma$  msg hops dip dsn dsk oip osn sip
  assumes "rreq_rrep_fresh (rt ( $\sigma$  sip)) (Rreq hops dip dsn dsk oip osn sip)"
  and "rreq_rrep_sn (Rreq hops dip dsn dsk oip osn sip)"
  shows "msg_fresh  $\sigma$  (Rreq hops dip dsn dsk oip osn sip)"
  (is "msg_fresh  $\sigma$  ?msg")
proof -
  let ?rt = "rt ( $\sigma$  sip)"
  from assms(2) have "1  $\leq$  osn" by simp
  thus ?thesis
  unfolding msg_fresh_def
  proof (simp only: msg.case, intro conjI impI)
    assume "sip  $\neq$  oip"
    with assms(1) show "oip  $\in$  kD(?rt)" by simp
  next
    assume "sip  $\neq$  oip"
    and "nsqn ?rt oip = osn"
    show "the (dhops ?rt oip)  $\leq$  hops  $\vee$  the (flag ?rt oip) = inv"
    proof (cases "oip  $\in$  vD(?rt)")
      assume "oip  $\in$  vD(?rt)"
      hence "nsqn ?rt oip = sqn ?rt oip" ..
      with  $\langle$ nsqn ?rt oip = osn $\rangle$  have "sqn ?rt oip = osn" by simp
      with assms(1) and  $\langle$ sip  $\neq$  oip $\rangle$  have "the (dhops ?rt oip)  $\leq$  hops"
      by simp
      thus ?thesis ..
    next
      assume "oip  $\notin$  vD(?rt)"
      moreover from assms(1) and  $\langle$ sip  $\neq$  oip $\rangle$  have "oip  $\in$  kD(?rt)" by simp
      ultimately have "oip  $\in$  iD(?rt)" by auto
      hence "the (flag ?rt oip) = inv" ..
      thus ?thesis ..
    qed
  next
    assume "sip  $\neq$  oip"
    with assms(1) have "osn  $\leq$  sqn ?rt oip" by auto
    thus "osn  $\leq$  nsqn (rt ( $\sigma$  sip)) oip"
    proof (rule nat_le_eq_or_lt)
      assume "osn < sqn ?rt oip"
      hence "osn  $\leq$  sqn ?rt oip - 1" by simp
      also have "...  $\leq$  nsqn ?rt oip" by (rule sqn_nsqn)
      finally show "osn  $\leq$  nsqn ?rt oip" .
    next
      assume "osn = sqn ?rt oip"
      with assms(1) and  $\langle$ sip  $\neq$  oip $\rangle$  have "oip  $\in$  kD(?rt)"
      and "the (flag ?rt oip) = val"
      by auto
      hence "nsqn ?rt oip = sqn ?rt oip" ..
      with  $\langle$ osn = sqn ?rt oip $\rangle$  have "nsqn ?rt oip = osn" by simp
      thus "osn  $\leq$  nsqn ?rt oip" by simp
    qed
  qed simp
qed

lemma rrep_nsqn_is_fresh [simp]:
  fixes  $\sigma$  msg hops dip dsn oip sip
  assumes "rreq_rrep_fresh (rt ( $\sigma$  sip)) (Rrep hops dip dsn oip sip)"
  and "rreq_rrep_sn (Rrep hops dip dsn oip sip)"
  shows "msg_fresh  $\sigma$  (Rrep hops dip dsn oip sip)"
  (is "msg_fresh  $\sigma$  ?msg")

```



```

proof -
  let ?rt = "rt ( $\sigma$  sip)"
  from assms have "sip  $\neq$  dip  $\longrightarrow$  dip $\in$ kD(?rt)  $\wedge$  sqn ?rt dip = dsn  $\wedge$  the (flag ?rt dip) = val"
    by simp
  hence "sip  $\neq$  dip  $\longrightarrow$  dip $\in$ kD(?rt)  $\wedge$  nsqn ?rt dip  $\geq$  dsn"
  by clarsimp
  with assms show "msg_fresh  $\sigma$  ?msg"
  by clarsimp
qed

```

lemma rerr\_nsqn\_is\_fresh [simp]:

```

fixes  $\sigma$  msg dests sip
assumes "rerr_invalid (rt ( $\sigma$  sip)) (Rerr dests sip)"
shows "msg_fresh  $\sigma$  (Rerr dests sip)"
  (is "msg_fresh  $\sigma$  ?msg")

```

proof -

```

let ?rt = "rt ( $\sigma$  sip)"
from assms have *: "( $\forall$ rip $\in$ dom(dests). (rip $\in$ iD(rt ( $\sigma$  sip))
   $\wedge$  the (dests rip) = sqn (rt ( $\sigma$  sip)) rip))"

```

```

  by clarsimp
have "( $\forall$ rip $\in$ dom(dests). (rip $\in$ kD(rt ( $\sigma$  sip))
   $\wedge$  the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip))"

```

proof

```

fix rip
assume "rip  $\in$  dom dests"
with * have "rip $\in$ iD(rt ( $\sigma$  sip))" and "the (dests rip) = sqn (rt ( $\sigma$  sip)) rip"
  by auto

```

```

from this(2) have "the (dests rip) - 1 = sqn (rt ( $\sigma$  sip)) rip - 1" by simp
also have "...  $\leq$  nsqn (rt ( $\sigma$  sip)) rip" by (rule sqn_nsqn)
finally have "the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip" .

```

```

with (rip $\in$ iD(rt ( $\sigma$  sip)))
  show "rip $\in$ kD(rt ( $\sigma$  sip))  $\wedge$  the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip"
  by clarsimp

```

qed

```

thus "msg_fresh  $\sigma$  ?msg"
  by simp

```

qed

lemma quality\_increases\_msg\_fresh [elim]:

```

assumes qinc: " $\forall$ j. quality_increases ( $\sigma$  j) ( $\sigma'$  j)"
  and "msg_fresh  $\sigma$  m"
shows "msg_fresh  $\sigma'$  m"

```

using assms(2)

proof (cases m)

```

fix hops rreqid dip dsn dsk oip osn sip
assume [simp]: "m = Rreq hops dip dsn dsk oip osn sip"
  and "msg_fresh  $\sigma$  m"
then have "osn  $\geq$  1" and "sip = oip  $\vee$  (oip $\in$ kD(rt ( $\sigma$  sip))  $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma$  sip)) oip
   $\wedge$  (nsqn (rt ( $\sigma$  sip)) oip = osn
   $\longrightarrow$  (the (dhops (rt ( $\sigma$  sip)) oip)  $\leq$  hops
   $\vee$  the (flag (rt ( $\sigma$  sip)) oip) = inv)))"

```

by auto

from this(2) show ?thesis

proof

```

assume "sip = oip" with (osn  $\geq$  1) show ?thesis by simp

```

next

```

assume "oip $\in$ kD(rt ( $\sigma$  sip))  $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma$  sip)) oip
   $\wedge$  (nsqn (rt ( $\sigma$  sip)) oip = osn
   $\longrightarrow$  (the (dhops (rt ( $\sigma$  sip)) oip)  $\leq$  hops
   $\vee$  the (flag (rt ( $\sigma$  sip)) oip) = inv))"

```

moreover from qinc have "quality\_increases ( $\sigma$  sip) ( $\sigma'$  sip)" ..

ultimately have "oip $\in$ kD(rt ( $\sigma'$  sip))  $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma'$  sip)) oip

```

       $\wedge$  (nsqn (rt ( $\sigma'$  sip)) oip = osn
         $\rightarrow$  (the (dhops (rt ( $\sigma'$  sip)) oip)  $\leq$  hops
           $\vee$  the (flag (rt ( $\sigma'$  sip)) oip) = inv))"
    using (osn  $\geq$  1) by (rule quality_increases_rreq_rrep_props [rotated 2])
    with (osn  $\geq$  1) show "msg_fresh  $\sigma'$  m"
      by (clarsimp)
  qed
next
fix hops dip dsn oip sip
assume [simp]: "m = Rrep hops dip dsn oip sip"
  and "msg_fresh  $\sigma$  m"
then have "dsn  $\geq$  1" and "sip = dip  $\vee$  (dip  $\in$  kD(rt ( $\sigma$  sip))  $\wedge$  dsn  $\leq$  nsqn (rt ( $\sigma$  sip)) dip
   $\wedge$  (nsqn (rt ( $\sigma$  sip)) dip = dsn
     $\rightarrow$  (the (dhops (rt ( $\sigma$  sip)) dip)  $\leq$  hops
       $\vee$  the (flag (rt ( $\sigma$  sip)) dip) = inv))"

  by auto
from this(2) show "?thesis"
proof
  assume "sip = dip" with (dsn  $\geq$  1) show ?thesis by simp
next
  assume "dip  $\in$  kD(rt ( $\sigma$  sip))  $\wedge$  dsn  $\leq$  nsqn (rt ( $\sigma$  sip)) dip
     $\wedge$  (nsqn (rt ( $\sigma$  sip)) dip = dsn
       $\rightarrow$  (the (dhops (rt ( $\sigma$  sip)) dip)  $\leq$  hops
         $\vee$  the (flag (rt ( $\sigma$  sip)) dip) = inv))"
  moreover from qinc have "quality_increases ( $\sigma$  sip) ( $\sigma'$  sip)" ..
  ultimately have "dip  $\in$  kD(rt ( $\sigma'$  sip))  $\wedge$  dsn  $\leq$  nsqn (rt ( $\sigma'$  sip)) dip
     $\wedge$  (nsqn (rt ( $\sigma'$  sip)) dip = dsn
       $\rightarrow$  (the (dhops (rt ( $\sigma'$  sip)) dip)  $\leq$  hops
         $\vee$  the (flag (rt ( $\sigma'$  sip)) dip) = inv))"

  using (dsn  $\geq$  1) by (rule quality_increases_rreq_rrep_props [rotated 2])
  with (dsn  $\geq$  1) show "msg_fresh  $\sigma'$  m"
    by clarsimp
  qed
next
fix dests sip
assume [simp]: "m = Rerr dests sip"
  and "msg_fresh  $\sigma$  m"
then have *: " $\forall$  rip  $\in$  dom(dests). rip  $\in$  kD(rt ( $\sigma$  sip))
   $\wedge$  the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip"

  by simp
have " $\forall$  rip  $\in$  dom(dests). rip  $\in$  kD(rt ( $\sigma'$  sip))
   $\wedge$  the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma'$  sip)) rip"

proof
  fix rip
  assume "rip  $\in$  dom(dests)"
  with * have "rip  $\in$  kD(rt ( $\sigma$  sip))" and "the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip"
    by - (drule(1) bspec, clarsimp)+
  moreover from qinc have "quality_increases ( $\sigma$  sip) ( $\sigma'$  sip)" by simp
  ultimately show "rip  $\in$  kD(rt ( $\sigma'$  sip))  $\wedge$  the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma'$  sip)) rip" ..
  qed
  thus ?thesis by simp
qed simp_all

```

end

## 1.8 The 'open' AODV model

```

theory A_OAodv
imports A_Aodv "../.../AWN/DAWN_SOS_Labels" "../.../AWN/DAWN_Convert"
begin

```

Definitions for stating and proving global network properties over individual processes.

```

definition  $\sigma_{AODV}'$  :: " $((ip \Rightarrow state) \times ((state, msg, pseqp, pseqp label) seqp)) set$ "
where " $\sigma_{AODV}' \equiv \{(\lambda i. aodv\_init\ i, \Gamma_{AODV}\ PAodv)\}$ "

```

```

abbreviation opaadv
  :: "ip  $\Rightarrow$  ((ip  $\Rightarrow$  state)  $\times$  (state, msg, pseq, pseq label) seqp, msg seq_action) automaton"
where
  "opaadv i  $\equiv$  ( $\lfloor$  init =  $\sigma_{AODV}'$ , trans = oseqp_sos  $\Gamma_{AODV}$  i  $\rfloor$ )"

lemma initiali_aadv [intro!, simp]: "initiali i (init (opaadv i)) (init (paadv i))"
  unfolding  $\sigma_{AODV\_def}$   $\sigma_{AODV}'\_def$  by rule simp_all

lemma oaadv_control_within [simp]: "control_within  $\Gamma_{AODV}$  (init (opaadv i))"
  unfolding  $\sigma_{AODV}'\_def$  by (rule control_withinI) (auto simp del:  $\Gamma_{AODV\_simps}$ )

lemma  $\sigma_{AODV}'\_labels$  [simp]: " $(\sigma, p) \in \sigma_{AODV}' \implies labels \Gamma_{AODV} p = \{PAadv-:0\}$ "
  unfolding  $\sigma_{AODV}'\_def$  by simp

lemma oaadv_init_kD_empty [simp]:
  " $(\sigma, p) \in \sigma_{AODV}' \implies kD (rt (\sigma i)) = \{\}$ "
  unfolding  $\sigma_{AODV}'\_def$  kD_def by simp

lemma oaadv_init_vD_empty [simp]:
  " $(\sigma, p) \in \sigma_{AODV}' \implies vD (rt (\sigma i)) = \{\}$ "
  unfolding  $\sigma_{AODV}'\_def$  vD_def by simp

lemma oaadv_trans: "trans (opaadv i) = oseqp_sos  $\Gamma_{AODV}$  i"
  by simp

declare
  oseq_invariant_ctermsI [OF aadv_wf oaadv_control_within aadv_simple_labels oaadv_trans, cterms_intros]
  oseq_step_invariant_ctermsI [OF aadv_wf oaadv_control_within aadv_simple_labels oaadv_trans, cterms_intros]

end

```

## 1.9 Global invariant proofs over sequential processes

```

theory A_Global_Invariants
imports A_Seq_Invariants
  A_Aadv_Predicates
  A_Fresher
  A_Quality_Increases
  "../.../AWN/OAWN_Convert"
  A_OAadv
begin

lemma other_quality_increases [elim]:
  assumes "other quality_increases I  $\sigma \sigma'$ "
  shows " $\forall j. quality\_increases (\sigma j) (\sigma' j)$ "
  using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
  fixes m
  assumes *: "otherwith P I (orecvmsg Q)  $\sigma \sigma' a$ "
    and weakenP: " $\bigwedge \sigma m. P \sigma m \implies P' \sigma m$ "
    and weakenQ: " $\bigwedge \sigma m. Q \sigma m \implies Q' \sigma m$ "
  shows "otherwith P' I (orecvmsg Q')  $\sigma \sigma' a$ "
proof
  fix j
  assume "j  $\notin I$ "
  with * have "P ( $\sigma j$ ) ( $\sigma' j$ )" by auto
  thus "P' ( $\sigma j$ ) ( $\sigma' j$ )" by (rule weakenP)
next
  from * have "orecvmsg Q  $\sigma a$ " by auto
  thus "orecvmsg Q'  $\sigma a$ "
  by rule (erule weakenQ)
qed

```

```

lemma oreceived_msg_inv:
  assumes other: " $\bigwedge \sigma \sigma' m. \llbracket P \sigma m; \text{other } Q \{i\} \sigma \sigma' \rrbracket \implies P \sigma' m$ "
  and local: " $\bigwedge \sigma m. P \sigma m \implies P (\sigma(i := \sigma i(\text{msg} := m))) m$ "
  shows "opaadv i  $\models$  (otherwith Q {i} (orecvmsg P), other Q {i}  $\rightarrow$ )
        onl  $\Gamma_{AODV} (\lambda(\sigma, l). l \in \{\text{PAadv-:1}\} \rightarrow P \sigma (\text{msg} (\sigma i)))$ "
proof (inv_cterms, intro impI)
  fix  $\sigma \sigma' l$ 
  assume "l = PAadv-:1  $\rightarrow$  P  $\sigma$  (msg ( $\sigma$  i))"
  and "l = PAadv-:1"
  and "other Q {i}  $\sigma \sigma'$ "
  from this(1-2) have "P  $\sigma$  (msg ( $\sigma$  i))" ..
  hence "P  $\sigma'$  (msg ( $\sigma$  i))" using (other Q {i}  $\sigma \sigma'$ )
  by (rule other)
  moreover from (other Q {i}  $\sigma \sigma'$ ) have " $\sigma' i = \sigma i$ " ..
  ultimately show "P  $\sigma'$  (msg ( $\sigma' i$ ))" by simp
next
  fix  $\sigma \sigma' \text{msg}$ 
  assume "otherwith Q {i} (orecvmsg P)  $\sigma \sigma'$  (receive msg)"
  and " $\sigma' i = \sigma i(\text{msg} := \text{msg})$ "
  from this(1) have "P  $\sigma \text{msg}$ "
  and " $\forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j)$ " by auto
  from this(1) have "P ( $\sigma(i := \sigma i(\text{msg} := \text{msg}))$ ) msg" by (rule local)
  thus "P  $\sigma' \text{msg}$ "
  proof (rule other)
    from ( $\sigma' i = \sigma i(\text{msg} := \text{msg})$ ) and ( $\forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j)$ )
    show "other Q {i} ( $\sigma(i := \sigma i(\text{msg} := \text{msg}))$ )  $\sigma'$ "
    by - (rule otherI, auto)
  qed
qed

```

(Equivalent to) Proposition 7.27

```

lemma local_quality_increases:
  "paadv i  $\models_A$  (recvmsg rreq_rrep_sn  $\rightarrow$ ) onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \text{quality\_increases } \xi \xi')$ "
proof (rule step_invariantI)
  fix s a s'
  assume sr: "s  $\in$  reachable (paadv i) (recvmsg rreq_rrep_sn)"
  and tr: "(s, a, s')  $\in$  trans (paadv i)"
  and rm: "recvmsg rreq_rrep_sn a"
  from sr have srTT: "s  $\in$  reachable (paadv i) TT" ..

  from route_tables_fresher sr tr rm
  have "onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \forall \text{dip} \in \text{kD} (\text{rt } \xi). \text{rt } \xi \sqsubseteq_{\text{dip}} \text{rt } \xi') (s, a, s')$ "
  by (rule step_invariantD)

  moreover from known_destinations_increase srTT tr TT_True
  have "onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \text{kD} (\text{rt } \xi) \subseteq \text{kD} (\text{rt } \xi')) (s, a, s')$ "
  by (rule step_invariantD)

  moreover from sqns_increase srTT tr TT_True
  have "onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \forall \text{ip}. \text{sqn} (\text{rt } \xi) \text{ip} \leq \text{sqn} (\text{rt } \xi') \text{ip}) (s, a, s')$ "
  by (rule step_invariantD)

  ultimately show "onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \text{quality\_increases } \xi \xi') (s, a, s')$ "
  unfolding onll_def by auto
qed

```

```

lemmas olocal_quality_increases =
  open_seq_step_invariant [OF local_quality_increases initiali_aadv oaadv_trans aadv_trans,
    simplified seqll_onll_swap]

```

```

lemma oquality_increases:
  "opaadv i  $\models_A$  (otherwith quality_increases {i} (orecvmsg ( $\lambda_. \text{rreq\_rrep\_sn}$ )),
    other quality_increases {i}  $\rightarrow$ )

```

```

      onll  $\Gamma_{AODV}$  ( $\lambda((\sigma, \_), \_, (\sigma', \_)). \forall j. \text{quality\_increases } (\sigma j) (\sigma' j)$ )"
(is " $\_ \models_A (?S, \_ \rightarrow) \_$ ")
proof (rule onll_ostep_invariantI, simp)
  fix  $\sigma p l a \sigma' p' l'$ 
  assume or: " $(\sigma, p) \in \text{oreachable } (\text{opaadv } i) ?S (\text{other quality\_increases } \{i\})$ "
    and  $ll: "l \in \text{labels } \Gamma_{AODV} p"$ 
    and " $?S \sigma \sigma' a$ "
    and  $tr: "((\sigma, p), a, (\sigma', p')) \in \text{oseqp\_sos } \Gamma_{AODV} i"$ 
    and  $ll': "l' \in \text{labels } \Gamma_{AODV} p'"$ 
  from this(1-3) have "orecvmsg ( $\lambda_. \text{rreq\_rrep\_sn}$ )  $\sigma a$ "
    by (auto dest!: oreachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"
      and QU="other quality_increases {i}]"
      otherwith_actionD)
  with or have orw: " $(\sigma, p) \in \text{oreachable } (\text{opaadv } i) (\text{act (recvmsg rreq\_rrep\_sn)})$ "
    (other quality_increases {i})"
    by - (erule oreachable_weakenE, auto)
  with  $tr ll ll'$  and (orecvmsg ( $\lambda_. \text{rreq\_rrep\_sn}$ )  $\sigma a$ ) have "quality_increases ( $\sigma i$ ) ( $\sigma' i$ )"
    by - (drule onll_ostep_invariantD [OF olocal_quality_increases], auto simp: seqll_def)
  with (?S  $\sigma \sigma' a$ ) show " $\forall j. \text{quality\_increases } (\sigma j) (\sigma' j)$ "
    by (auto dest!: otherwith_syncD)
qed

```

lemma rreq\_rrep\_nsqn\_fresh\_any\_step\_invariant:

```

"opaadv  $i \models_A (\text{act (recvmsg rreq\_rrep\_sn)}, \text{other } A \{i\} \rightarrow)$ "
  onll  $\Gamma_{AODV}$  ( $\lambda((\sigma, \_), a, \_). \text{anycast (msg\_fresh } \sigma) a$ )"
proof (rule ostep_invariantI, simp del: act_simp)
  fix  $\sigma p a \sigma' p'$ 
  assume or: " $(\sigma, p) \in \text{oreachable } (\text{opaadv } i) (\text{act (recvmsg rreq\_rrep\_sn)}) (\text{other } A \{i\})$ "
    and " $((\sigma, p), a, (\sigma', p')) \in \text{oseqp\_sos } \Gamma_{AODV} i$ "
    and  $recv: "act (recvmsg rreq\_rrep\_sn) \sigma \sigma' a"$ 
  obtain  $l l'$  where " $l \in \text{labels } \Gamma_{AODV} p$ " and " $l' \in \text{labels } \Gamma_{AODV} p'$ "
    by (metis aadv_ex_label)
  from  $((\sigma, p), a, (\sigma', p')) \in \text{oseqp\_sos } \Gamma_{AODV} i$ 
    have  $tr: "((\sigma, p), a, (\sigma', p')) \in \text{trans } (\text{opaadv } i)"$  by simp
  have "anycast (rreq_rrep_fresh (rt ( $\sigma i$ )))  $a$ "
  proof -
    have "opaadv  $i \models_A (\text{act (recvmsg rreq\_rrep\_sn)}, \text{other } A \{i\} \rightarrow)$ "
      onll  $\Gamma_{AODV}$  (seqll  $i (\lambda((\xi, \_), a, \_). \text{anycast (rreq\_rrep\_fresh (rt } \xi)) a)$ )"
    by (rule ostep_invariant_weakenE [OF
      open_seq_step_invariant [OF rreq_rrep_fresh_any_step_invariant initiali_aadv,
        simplified seqll_onll_swap]]) auto
    hence "onll  $\Gamma_{AODV}$  (seqll  $i (\lambda((\xi, \_), a, \_). \text{anycast (rreq\_rrep\_fresh (rt } \xi)) a)$ )"
      ( $((\sigma, p), a, (\sigma', p'))$ )"
    using or  $tr$   $recv$  by - (erule(4) ostep_invariantE)
    thus ?thesis
      using ( $l \in \text{labels } \Gamma_{AODV} p$ ) and ( $l' \in \text{labels } \Gamma_{AODV} p'$ ) by auto
  qed

```

moreover have "anycast (rerr\_invalid (rt ( $\sigma i$ )))  $a$ "

```

proof -
  have "opaadv  $i \models_A (\text{act (recvmsg rreq\_rrep\_sn)}, \text{other } A \{i\} \rightarrow)$ "
    onll  $\Gamma_{AODV}$  (seqll  $i (\lambda((\xi, \_), a, \_). \text{anycast (rerr\_invalid (rt } \xi)) a)$ )"
  by (rule ostep_invariant_weakenE [OF
    open_seq_step_invariant [OF rerr_invalid_any_step_invariant initiali_aadv,
      simplified seqll_onll_swap]]) auto
  hence "onll  $\Gamma_{AODV}$  (seqll  $i (\lambda((\xi, \_), a, \_). \text{anycast (rerr\_invalid (rt } \xi)) a)$ )"
    ( $((\sigma, p), a, (\sigma', p'))$ )"
  using or  $tr$   $recv$  by - (erule(4) ostep_invariantE)
  thus ?thesis
    using ( $l \in \text{labels } \Gamma_{AODV} p$ ) and ( $l' \in \text{labels } \Gamma_{AODV} p'$ ) by auto
qed

```

moreover have "anycast rreq\_rrep\_sn  $a$ "

```

proof -
  from or tr recv
  have "onll  $\Gamma_{AODV}$  (seqll i ( $\lambda(\_, a, \_)$ . anycast rreq_rrep_sn a)) (( $\sigma$ , p), a, ( $\sigma'$ , p'))"
  by (rule ostep_invariantE [OF
    open_seq_step_invariant [OF rreq_rrep_sn_any_step_invariant initiali_aodv
      oadv_trans aadv_trans,
      simplified seqll_onll_swap]])
  thus ?thesis
  using ( $l \in \text{labels } \Gamma_{AODV} p$ ) and ( $l' \in \text{labels } \Gamma_{AODV} p'$ ) by auto
qed

moreover have "anycast ( $\lambda m$ . not_Pkt m  $\longrightarrow$  msg_sender m = i) a"
proof -
  have "opaadv i  $\models_A$  (act (recvmmsg rreq_rrep_sn), other A {i}  $\rightarrow$ )
    onll  $\Gamma_{AODV}$  (seqll i ( $\lambda(\xi, \_)$ , a,  $\_)$ . anycast ( $\lambda m$ . not_Pkt m  $\longrightarrow$  msg_sender m = i) a))"
  by (rule ostep_invariant_weakenE [OF
    open_seq_step_invariant [OF sender_ip_valid initiali_aodv,
      simplified seqll_onll_swap]]) auto
  thus ?thesis using or tr recv ( $l \in \text{labels } \Gamma_{AODV} p$ ) and ( $l' \in \text{labels } \Gamma_{AODV} p'$ )
  by - (drule(3) onll_ostep_invariantD, auto)
qed

ultimately have "anycast (msg_fresh  $\sigma$ ) a"
by (simp_all add: anycast_def
  del: msg_fresh
  split: seq_action.split_asm msg.split_asm) simp_all
thus "onll  $\Gamma_{AODV}$  ( $\lambda((\sigma, \_)$ , a,  $\_)$ . anycast (msg_fresh  $\sigma$ ) a) (( $\sigma$ , p), a, ( $\sigma'$ , p'))"
by auto
qed

lemma oreceived_rreq_rrep_nsqn_fresh_inv:
"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onll  $\Gamma_{AODV}$  ( $\lambda(\sigma, l)$ .  $l \in \{PAodv-:1\} \longrightarrow$  msg_fresh  $\sigma$  (msg ( $\sigma$  i)))"
proof (rule oreceived_msg_inv)
  fix  $\sigma \sigma' m$ 
  assume *: "msg_fresh  $\sigma$  m"
  and "other quality_increases {i}  $\sigma \sigma'$ "
  from this(2) have " $\forall j$ . quality_increases ( $\sigma$  j) ( $\sigma'$  j)" ..
  thus "msg_fresh  $\sigma' m$ " using * ..
next
  fix  $\sigma m$ 
  assume "msg_fresh  $\sigma$  m"
  thus "msg_fresh ( $\sigma$ (i :=  $\sigma$  i (msg := m))) m"
  proof (cases m)
    fix dests sip
    assume "m = Rerr dests sip"
    with (msg_fresh  $\sigma$  m) show ?thesis by auto
  qed auto
qed

lemma oquality_increases_nsqn_fresh:
"opaadv i  $\models_A$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onll  $\Gamma_{AODV}$  ( $\lambda((\sigma, \_)$ ,  $\_)$ , ( $\sigma'$ ,  $\_)$ ).  $\forall j$ . quality_increases ( $\sigma$  j) ( $\sigma'$  j))"
by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onll  $\Gamma_{AODV}$  (seq1 i ( $\lambda(\xi, l)$ .  $l \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-:n \mid n. \text{True}\} \longrightarrow 1 \leq \text{osn } \xi$ ))"
by (rule oinvariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_aodv]])
(auto simp: seq1_onll_swap)

```

lemma rreq\_sip:

```

"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onl  $\Gamma_{AODV}$  ( $\lambda(\sigma, l)$ .
    ( $l \in \{PAadv:-4, PAadv:-5, PRreq:-0, PRreq:-2\} \wedge sip(\sigma i) \neq oip(\sigma i)$ 
       $\rightarrow oip(\sigma i) \in kD(rt(\sigma(sip(\sigma i))))$ 
         $\wedge nsqn(rt(\sigma(sip(\sigma i))))(oip(\sigma i)) \geq osn(\sigma i)$ 
         $\wedge (nsqn(rt(\sigma(sip(\sigma i))))(oip(\sigma i)) = osn(\sigma i)$ 
           $\rightarrow (hops(\sigma i) \geq the(dhops(rt(\sigma(sip(\sigma i))))(oip(\sigma i)))$ 
             $\vee the(flag(rt(\sigma(sip(\sigma i))))(oip(\sigma i))) = inv)))$ "
(is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")
proof (inv_terms inv add: oseq_step_invariant_terms [OF oquality_increases_nsqn_fresh
  aadv_wf oaadv_trans]
  onl_oinvariant_terms [OF aadv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
  onl_oinvariant_terms [OF aadv_wf oosn_rreq]
  simp add: seqsimp
  simp del: One_nat_def, rule impI)
fix  $\sigma \sigma' p l$ 
assume " $(\sigma, p) \in oreachable(opaadv i) ?S ?U$ "
and " $l \in labels \Gamma_{AODV} p$ "
and pre:
  " $(l = PAadv:-4 \vee l = PAadv:-5 \vee l = PRreq:-0 \vee l = PRreq:-2) \wedge sip(\sigma i) \neq oip(\sigma i)$ 
 $\rightarrow oip(\sigma i) \in kD(rt(\sigma(sip(\sigma i))))$ 
 $\wedge osn(\sigma i) \leq nsqn(rt(\sigma(sip(\sigma i))))(oip(\sigma i))$ 
 $\wedge (nsqn(rt(\sigma(sip(\sigma i))))(oip(\sigma i)) = osn(\sigma i)$ 
 $\rightarrow the(dhops(rt(\sigma(sip(\sigma i))))(oip(\sigma i))) \leq hops(\sigma i)$ 
 $\vee the(flag(rt(\sigma(sip(\sigma i))))(oip(\sigma i))) = inv)"$ 
and "other quality_increases {i}  $\sigma \sigma'$ "
and hyp: " $(l=PAadv:-4 \vee l=PAadv:-5 \vee l=PRreq:-0 \vee l=PRreq:-2) \wedge sip(\sigma' i) \neq oip(\sigma' i)$ "
(is "?labels  $\wedge sip(\sigma' i) \neq oip(\sigma' i)$ ")
from this(4) have " $\sigma' i = \sigma i$ " ..
with hyp have hyp': "?labels  $\wedge sip(\sigma i) \neq oip(\sigma i)$ " by simp
show " $oip(\sigma' i) \in kD(rt(\sigma'(sip(\sigma' i))))$ "
 $\wedge osn(\sigma' i) \leq nsqn(rt(\sigma'(sip(\sigma' i))))(oip(\sigma' i))$ 
 $\wedge (nsqn(rt(\sigma'(sip(\sigma' i))))(oip(\sigma' i)) = osn(\sigma' i)$ 
 $\rightarrow the(dhops(rt(\sigma'(sip(\sigma' i))))(oip(\sigma' i))) \leq hops(\sigma' i)$ 
 $\vee the(flag(rt(\sigma'(sip(\sigma' i))))(oip(\sigma' i))) = inv)"$ 
proof (cases " $sip(\sigma i) = i$ ")
assume " $sip(\sigma i) \neq i$ "
from (other quality_increases {i}  $\sigma \sigma'$ )
  have "quality_increases ( $\sigma(sip(\sigma i))$ ) ( $\sigma'(sip(\sigma' i))$ )"
  by (rule otherE) (clarsimp simp: (sip  $\sigma i \neq i$ ))
moreover from ( $(\sigma, p) \in oreachable(opaadv i) ?S ?U$ ) ( $l \in labels \Gamma_{AODV} p$ ) and hyp
  have " $l \leq osn(\sigma' i)$ "
  by (auto dest!: onl_oinvariant_weakenD [OF oosn_rreq]
    simp add: seqsimp (sip  $\sigma i = \sigma i$ ))
moreover from (sip  $\sigma i \neq i$ ) hyp' and pre
  have " $oip(\sigma' i) \in kD(rt(\sigma(sip(\sigma i))))$ "
 $\wedge osn(\sigma' i) \leq nsqn(rt(\sigma(sip(\sigma i))))(oip(\sigma' i))$ 
 $\wedge (nsqn(rt(\sigma(sip(\sigma i))))(oip(\sigma' i)) = osn(\sigma' i)$ 
 $\rightarrow the(dhops(rt(\sigma(sip(\sigma i))))(oip(\sigma' i))) \leq hops(\sigma' i)$ 
 $\vee the(flag(rt(\sigma(sip(\sigma i))))(oip(\sigma' i))) = inv)"$ 
  by (auto simp: (sip  $\sigma i = \sigma i$ ))
ultimately show ?thesis
  by (rule quality_increases_rreq_rrep_props)
next
assume " $sip(\sigma i) = i$ " thus ?thesis
  using (sip  $\sigma i = \sigma i$ ) hyp and pre by auto
qed
qed (auto elim!: quality_increases_rreq_rrep_props')

```

lemma odsn\_rrep:

```

"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )

```

$\text{onl } \Gamma_{AODV} (\text{seq1 } i (\lambda(\xi, l). l \in \{\text{PAadv-:6, PAadv-:7}\} \cup \{\text{PRrep-:n/n. True}\} \rightarrow 1 \leq \text{dsn } \xi))$ "  
 by (rule oinvariant\_weakenE [OF open\_seq\_invariant [OF dsn\_rrep initiali\_adv]])  
 (auto simp: seq1\_onl\_swap)

lemma rrep\_sip:

"opaadv i  $\models$  (otherwith quality\_increases {i} (orecvmsg msg\_fresh),  
 other quality\_increases {i}  $\rightarrow$ )  
 onl  $\Gamma_{AODV} (\lambda(\sigma, l).$   
 $(l \in \{\text{PAadv-:6, PAadv-:7, PRrep-:0, PRrep-:1}\} \wedge \text{sip } (\sigma i) \neq \text{dip } (\sigma i))$   
 $\rightarrow \text{dip } (\sigma i) \in \text{kD}(\text{rt } (\sigma (\text{sip } (\sigma i))))$   
 $\wedge \text{nsqn } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i)) \geq \text{dsn } (\sigma i)$   
 $\wedge (\text{nsqn } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i)) = \text{dsn } (\sigma i))$   
 $\rightarrow (\text{hops } (\sigma i) \geq \text{the } (\text{dhops } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i))))$   
 $\vee \text{the } (\text{flag } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i))) = \text{inv}))$ "  
 (is "\_  $\models$  (?S, ?U  $\rightarrow$ ) \_")  
 proof (inv\_terms inv add: oseq\_step\_invariant\_sterms [OF oquality\_increases\_nsqn\_fresh adv\_wf  
 oadv\_trans]  
 onl\_oinvariant\_sterms [OF adv\_wf oreceived\_rreq\_rrep\_nsqn\_fresh\_inv]  
 onl\_oinvariant\_sterms [OF adv\_wf odsn\_rrep]  
 simp del: One\_nat\_def, rule impI)  
 fix  $\sigma \sigma' p l$   
 assume "( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U"  
 and " $l \in \text{labels } \Gamma_{AODV} p$ "  
 and pre:  
 "( $l = \text{PAadv-:6} \vee l = \text{PAadv-:7} \vee l = \text{PRrep-:0} \vee l = \text{PRrep-:1}$ )  $\wedge \text{sip } (\sigma i) \neq \text{dip } (\sigma i)$   
 $\rightarrow \text{dip } (\sigma i) \in \text{kD} (\text{rt } (\sigma (\text{sip } (\sigma i))))$   
 $\wedge \text{dsn } (\sigma i) \leq \text{nsqn } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i))$   
 $\wedge (\text{nsqn } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i)) = \text{dsn } (\sigma i))$   
 $\rightarrow \text{the } (\text{dhops } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i))) \leq \text{hops } (\sigma i)$   
 $\vee \text{the } (\text{flag } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i))) = \text{inv}"$   
 and "other quality\_increases {i}  $\sigma \sigma'$ "  
 and hyp: "( $l = \text{PAadv-:6} \vee l = \text{PAadv-:7} \vee l = \text{PRrep-:0} \vee l = \text{PRrep-:1}$ )  $\wedge \text{sip } (\sigma' i) \neq \text{dip } (\sigma' i)$ "  
 (is "?labels  $\wedge \text{sip } (\sigma' i) \neq \text{dip } (\sigma' i)$ ")  
 from this(4) have " $\sigma' i = \sigma i$ " ..  
 with hyp have hyp': "?labels  $\wedge \text{sip } (\sigma i) \neq \text{dip } (\sigma i)$ " by simp  
 show " $\text{dip } (\sigma' i) \in \text{kD} (\text{rt } (\sigma' (\text{sip } (\sigma' i))))$   
 $\wedge \text{dsn } (\sigma' i) \leq \text{nsqn } (\text{rt } (\sigma' (\text{sip } (\sigma' i)))) (\text{dip } (\sigma' i))$   
 $\wedge (\text{nsqn } (\text{rt } (\sigma' (\text{sip } (\sigma' i)))) (\text{dip } (\sigma' i)) = \text{dsn } (\sigma' i))$   
 $\rightarrow \text{the } (\text{dhops } (\text{rt } (\sigma' (\text{sip } (\sigma' i)))) (\text{dip } (\sigma' i))) \leq \text{hops } (\sigma' i)$   
 $\vee \text{the } (\text{flag } (\text{rt } (\sigma' (\text{sip } (\sigma' i)))) (\text{dip } (\sigma' i))) = \text{inv}"$   
 proof (cases " $\text{sip } (\sigma i) = i$ ")  
 assume " $\text{sip } (\sigma i) \neq i$ "  
 from (other quality\_increases {i}  $\sigma \sigma'$ )  
 have "quality\_increases ( $\sigma (\text{sip } (\sigma i))$ ) ( $\sigma' (\text{sip } (\sigma' i))$ )"  
 by (rule otherE) (clarsimp simp:  $\text{sip } (\sigma i) \neq i$ )  
 moreover from ( $(\sigma, p) \in \text{oreachable } (\text{opaadv } i) ?S ?U$ ) ( $l \in \text{labels } \Gamma_{AODV} p$ ) and hyp  
 have " $1 \leq \text{dsn } (\sigma' i)$ "  
 by (auto dest!: onl\_oinvariant\_weakenD [OF odsn\_rrep]  
 simp add: seq1simp ( $\sigma' i = \sigma i$ ))  
 moreover from ( $\text{sip } (\sigma i) \neq i$ ) hyp' and pre  
 have " $\text{dip } (\sigma' i) \in \text{kD} (\text{rt } (\sigma (\text{sip } (\sigma i))))$   
 $\wedge \text{dsn } (\sigma' i) \leq \text{nsqn } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma' i))$   
 $\wedge (\text{nsqn } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma' i)) = \text{dsn } (\sigma' i))$   
 $\rightarrow \text{the } (\text{dhops } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma' i))) \leq \text{hops } (\sigma' i)$   
 $\vee \text{the } (\text{flag } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma' i))) = \text{inv}"$   
 by (auto simp: ( $\sigma' i = \sigma i$ ))  
 ultimately show ?thesis  
 by (rule quality\_increases\_rreq\_rrep\_props)  
 next  
 assume " $\text{sip } (\sigma i) = i$ " thus ?thesis  
 using ( $\sigma' i = \sigma i$ ) hyp and pre by auto  
 qed  
 qed (auto simp add: seq1simp elim!: quality\_increases\_rreq\_rrep\_props')



lemma rerr\_sip:

```
"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onl  $\Gamma_{AODV}$  ( $\lambda(\sigma, l).$ 
    l  $\in$  {PAadv-:8, PAadv-:9, PRerr-:0, PRerr-:1}
     $\rightarrow$  ( $\forall ripc \in \text{dom}(\text{dests } (\sigma i)). ripc \in kD(\text{rt } (\sigma (\text{sip } (\sigma i)))) \wedge$ 
      the (dests ( $\sigma i$ ) ripc) - 1  $\leq$  nsqn (rt ( $\sigma (\text{sip } (\sigma i))))$  ripc))"
(is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")
```

proof -

```
{ fix dests rip sip rsn and  $\sigma \sigma' ::$  "ip  $\Rightarrow$  state"
  assume qinc: " $\forall j.$  quality_increases ( $\sigma j$ ) ( $\sigma' j$ )"
  and *: " $\forall rip \in \text{dom } \text{dests}. rip \in kD(\text{rt } (\sigma \text{ sip}))$ 
     $\wedge$  the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma \text{ sip}$ )) rip"
  and "dests rip = Some rsn"
```

from this(3) have "rip  $\in$  dom dests" by auto

with \* and  $\langle$ dests rip = Some rsn $\rangle$  have "rip  $\in$  kD(rt ( $\sigma \text{ sip}$ ))"

and "rsn - 1  $\leq$  nsqn (rt ( $\sigma \text{ sip}$ )) rip"

by (auto dest!: bspec)

from qinc have "quality\_increases ( $\sigma \text{ sip}$ ) ( $\sigma' \text{ sip}$ )" ..

have "rip  $\in$  kD(rt ( $\sigma' \text{ sip}$ ))  $\wedge$  rsn - 1  $\leq$  nsqn (rt ( $\sigma' \text{ sip}$ )) rip"

proof

from  $\langle$ rip  $\in$  kD(rt ( $\sigma \text{ sip}$ )) $\rangle$  and  $\langle$ quality\_increases ( $\sigma \text{ sip}$ ) ( $\sigma' \text{ sip}$ ) $\rangle$

show "rip  $\in$  kD(rt ( $\sigma' \text{ sip}$ ))" ..

next

from  $\langle$ rip  $\in$  kD(rt ( $\sigma \text{ sip}$ )) $\rangle$  and  $\langle$ quality\_increases ( $\sigma \text{ sip}$ ) ( $\sigma' \text{ sip}$ ) $\rangle$

have "nsqn (rt ( $\sigma \text{ sip}$ )) rip  $\leq$  nsqn (rt ( $\sigma' \text{ sip}$ )) rip" ..

with  $\langle$ rsn - 1  $\leq$  nsqn (rt ( $\sigma \text{ sip}$ )) rip $\rangle$  show "rsn - 1  $\leq$  nsqn (rt ( $\sigma' \text{ sip}$ )) rip"

by (rule le\_trans)

qed

} note partial = this

show ?thesis

by (inv\_cterms inv add: oseq\_step\_invariant\_sterms [OF oquality\_increases\_nsqn\_fresh aadv\_wf  
oaadv\_trans]

onl\_oinvariant\_sterms [OF aadv\_wf oreceived\_rreq\_rrep\_nsqn\_fresh\_inv]

other\_quality\_increases other\_localD

simp del: One\_nat\_def, intro conjI)

(clarsimp simp del: One\_nat\_def split: split\_if\_asm option.split\_asm, erule(2) partial)+

qed

lemma prerr\_guard: "paadv i  $\models$

onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l).$  (l = PRerr-:1

$\rightarrow$  ( $\forall ip \in \text{dom}(\text{dests } \xi). ip \in vD(\text{rt } \xi)$

$\wedge$  the (nhop (rt  $\xi$ ) ip) = sip  $\xi$

$\wedge$  sqn (rt  $\xi$ ) ip < the (dests  $\xi$  ip))))"

by (inv\_cterms) (clarsimp split: option.split\_asm split\_if\_asm)

lemmas oaddpreRT\_welldefined =

open\_seq\_invariant [OF addpreRT\_welldefined initiali\_aadv oaadv\_trans aadv\_trans,

simplified seql\_onl\_swap,

THEN oinvariant\_anyact]

lemmas odests\_vD\_inc\_sqn =

open\_seq\_invariant [OF dests\_vD\_inc\_sqn initiali\_aadv oaadv\_trans aadv\_trans,

simplified seql\_onl\_swap,

THEN oinvariant\_anyact]

lemmas oprerr\_guard =

open\_seq\_invariant [OF prerr\_guard initiali\_aadv oaadv\_trans aadv\_trans,

simplified seql\_onl\_swap,

THEN oinvariant\_anyact]

Proposition 7.28

lemma seq\_compare\_next\_hop':

```

"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$  onl  $\Gamma_{AODV}$  ( $\lambda(\sigma, \_)$ .
     $\forall$ dip. let nhop = the (nhop (rt ( $\sigma$  i)) dip)
      in dip  $\in$  kD(rt ( $\sigma$  i))  $\wedge$  nhop  $\neq$  dip  $\rightarrow$ 
        dip  $\in$  kD(rt ( $\sigma$  nhop))  $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma$  nhop)) dip)"
(is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")
proof -

{ fix nhop and  $\sigma$   $\sigma'$  :: "ip  $\Rightarrow$  state"
  assume pre: " $\forall$ dip $\in$ kD(rt ( $\sigma$  i)). nhop dip  $\neq$  dip  $\rightarrow$ 
    dip $\in$ kD(rt ( $\sigma$  (nhop dip)))  $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma$  (nhop dip))) dip"
    and qinc: " $\forall$ j. quality_increases ( $\sigma$  j) ( $\sigma'$  j)"
  have " $\forall$ dip $\in$ kD(rt ( $\sigma$  i)). nhop dip  $\neq$  dip  $\rightarrow$ 
    dip $\in$ kD(rt ( $\sigma'$  (nhop dip)))  $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
  proof (intro ballI impI)
    fix dip
    assume "dip $\in$ kD(rt ( $\sigma$  i))"
      and "nhop dip  $\neq$  dip"
    with pre have "dip $\in$ kD(rt ( $\sigma$  (nhop dip)))"
      and "nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma$  (nhop dip))) dip"
    by auto
    from qinc have qinc_nhop: "quality_increases ( $\sigma$  (nhop dip)) ( $\sigma'$  (nhop dip))" ..
    with  $\langle$ dip $\in$ kD(rt ( $\sigma$  (nhop dip))) $\rangle$  have "dip $\in$ kD (rt ( $\sigma'$  (nhop dip)))" ..

    moreover have "nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
  proof -
    from  $\langle$ dip $\in$ kD(rt ( $\sigma$  (nhop dip))) $\rangle$  qinc_nhop
      have "nsqn (rt ( $\sigma$  (nhop dip))) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip" ..
    with  $\langle$ nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma$  (nhop dip))) dip $\rangle$  show ?thesis
      by simp
  qed

  ultimately show "dip $\in$ kD(rt ( $\sigma'$  (nhop dip)))
     $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip" ..
  qed
} note basic = this

{ fix nhop and  $\sigma$   $\sigma'$  :: "ip  $\Rightarrow$  state"
  assume pre: " $\forall$ dip $\in$ kD(rt ( $\sigma$  i)). nhop dip  $\neq$  dip  $\rightarrow$  dip $\in$ kD(rt ( $\sigma$  (nhop dip)))
     $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma$  (nhop dip))) dip"
    and ndest: " $\forall$ ripc $\in$ dom (dests ( $\sigma$  i)). ripc  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
     $\wedge$  the (dests ( $\sigma$  i) ripc) - 1  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) ripc"
    and issip: " $\forall$ ip $\in$ dom (dests ( $\sigma$  i)). nhop ip = sip ( $\sigma$  i)"
    and qinc: " $\forall$ j. quality_increases ( $\sigma$  j) ( $\sigma'$  j)"
  have " $\forall$ dip $\in$ kD(rt ( $\sigma$  i)). nhop dip  $\neq$  dip  $\rightarrow$  dip  $\in$  kD (rt ( $\sigma'$  (nhop dip)))
     $\wedge$  nsqn (invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i))) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
  proof (intro ballI impI)
    fix dip
    assume "dip $\in$ kD(rt ( $\sigma$  i))"
      and "nhop dip  $\neq$  dip"
    with pre and qinc have "dip $\in$ kD(rt ( $\sigma'$  (nhop dip)))"
      and "nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
    by (auto dest!: basic)

    have "nsqn (invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i))) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
  proof (cases "dip $\in$ dom (dests ( $\sigma$  i))")
    assume "dip $\in$ dom (dests ( $\sigma$  i))"
    with  $\langle$ dip $\in$ kD(rt ( $\sigma$  i)) $\rangle$  obtain dsn where "dests ( $\sigma$  i) dip = Some dsn"
      by auto
    with  $\langle$ dip $\in$ kD(rt ( $\sigma$  i)) $\rangle$  have "nsqn (invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i))) dip = dsn - 1"
      by (rule nsqn_invalidate_eq)
    moreover have "dsn - 1  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
  proof -
    from  $\langle$ dests ( $\sigma$  i) dip = Some dsn $\rangle$  have "the (dests ( $\sigma$  i) dip) = dsn" by simp
  }
}

```

```

with ndest and ⟨dip∈dom (dests (σ i))⟩ have "dip ∈ kD (rt (σ (sip (σ i))))"
                                                    "dsn - 1 ≤ nsqn (rt (σ (sip (σ i)))) dip"

  by auto
  moreover from issip and ⟨dip∈dom (dests (σ i))⟩ have "nhop dip = sip (σ i)" ..
  ultimately have "dip ∈ kD (rt (σ (nhop dip)))"
                and "dsn - 1 ≤ nsqn (rt (σ (nhop dip))) dip" by auto
  with qinc show "dsn - 1 ≤ nsqn (rt (σ' (nhop dip))) dip"
    by simp (metis kD_nsqn_quality_increases_trans)
qed
ultimately show ?thesis by simp
next
assume "dip ∉ dom (dests (σ i))"
with ⟨dip∈kD(rt (σ i))⟩
  have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = nsqn (rt (σ i)) dip"
    by (rule nsqn_invalidate_other)
  with ⟨nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip⟩ show ?thesis by simp
qed
with ⟨dip∈kD(rt (σ' (nhop dip)))⟩
  show "dip ∈ kD (rt (σ' (nhop dip)))"
    ∧ nsqn (invalidate (rt (σ i)) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ..
qed
} note basic_prerr = this

{ fix σ σ' :: "ip ⇒ state"
  assume a1: "∀dip∈kD(rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
            → dip∈kD(rt (σ (the (nhop (rt (σ i)) dip))))
            ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
  and a2: "∀j. quality_increases (σ j) (σ' j)"
  have "∀dip∈kD(rt (σ i)).
    the (nhop (update (rt (σ i)) (sip (σ i)) (0, unk, val, Suc 0, sip (σ i), {})) dip) ≠ dip →
    dip∈kD(rt (σ' (the (nhop (update (rt (σ i)) (sip (σ i))
      (0, unk, val, Suc 0, sip (σ i), {}))
      dip)))) ∧
    nsqn (update (rt (σ i)) (sip (σ i)) (0, unk, val, Suc 0, sip (σ i), {})) dip
    ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) (sip (σ i))
      (0, unk, val, Suc 0, sip (σ i), {}))
      dip))))
    dip" (is "∀dip∈kD(rt (σ i)). ?P dip")
proof
  fix dip
  assume "dip∈kD(rt (σ i))"
  with a1 and a2
    have "the (nhop (rt (σ i)) dip) ≠ dip → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
        ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
      by - (drule(1) basic, auto)
    thus "?P dip" by (cases "dip = sip (σ i)") auto
qed
} note nhop_update_sip = this

{ fix σ σ' oip sip osn hops
  assume pre: "∀dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
            → dip∈kD(rt (σ (the (nhop (rt (σ i)) dip))))
            ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
  and qinc: "∀j. quality_increases (σ j) (σ' j)"
  and *: "sip ≠ oip → oip∈kD(rt (σ sip))
        ∧ osn ≤ nsqn (rt (σ sip)) oip
        ∧ (nsqn (rt (σ sip)) oip = osn
          → the (dhops (rt (σ sip)) oip) ≤ hops
            ∨ the (flag (rt (σ sip)) oip) = inv)"
  from pre and qinc
  have pre': "∀dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
            → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
            ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
    by (rule basic)

```

```

have "(the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }))) oip) ≠ oip
  → oip ∈ kD(rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, { }))) oip))))
  ∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }))) oip
  ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, { }))) oip)))) oip)"
(is "?nhop_not_oip → ?oip_in_kD ∧ ?nsqn_le_nsqn")
proof (rule, split update_rt_split_asm)
  assume "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })"
  and "the (nhop (rt (σ i)) oip) ≠ oip"
  with pre' show "?oip_in_kD ∧ ?nsqn_le_nsqn" by auto
next
assume rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })"
  and notoip: ?nhop_not_oip
with * qinc have ?oip_in_kD
  by (clarsimp elim!: kD_quality_increases)
moreover with * pre qinc rtnot notoip have ?nsqn_le_nsqn
  by simp (metis kD_nsqn_quality_increases_trans)
ultimately show "?oip_in_kD ∧ ?nsqn_le_nsqn" ..
qed
} note update1 = this

{ fix σ σ' oip sip osn hops
  assume pre: "∀ dip ∈ kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip ∈ kD(rt (σ (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
  and qinc: "∀ j. quality_increases (σ j) (σ' j)"
  and *: "sip ≠ oip → oip ∈ kD(rt (σ sip))
    ∧ osn ≤ nsqn (rt (σ sip)) oip
    ∧ (nsqn (rt (σ sip)) oip = osn
      → the (dhops (rt (σ sip)) oip) ≤ hops
        ∨ the (flag (rt (σ sip)) oip) = inv)"
from pre and qinc
  have pre': "∀ dip ∈ kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip ∈ kD(rt (σ' (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
  by (rule basic)
have "∀ dip ∈ kD(rt (σ i)).
  the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }))) dip) ≠ dip
  → dip ∈ kD(rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, { }))) dip))))
  ∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }))) dip
  ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, { }))) dip)))) dip"
(is "∀ dip ∈ kD(rt (σ i)). _ → ?dip_in_kD dip ∧ ?nsqn_le_nsqn dip")
proof (intro ballI impI, split update_rt_split_asm)
  fix dip
  assume "dip ∈ kD(rt (σ i))"
  and "the (nhop (rt (σ i)) dip) ≠ dip"
  and "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })"
  with pre' show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip" by simp
next
  fix dip
  assume "dip ∈ kD(rt (σ i))"
  and notdip: "the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, { }))) dip) ≠ dip"
  and rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })"
  show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip"
  proof (cases "dip = oip")
    assume "dip ≠ oip"
    with pre' (dip ∈ kD(rt (σ i))) notdip
      show ?thesis by clarsimp
  next
    assume "dip = oip"

```

```

with rtnot qinc ⟨dip∈kD(rt (σ i))⟩ notdip *
  have "?dip_in_kD dip"
    by simp (metis kD_quality_increases)
moreover from ⟨dip = oip⟩ rtnot qinc ⟨dip∈kD(rt (σ i))⟩ notdip *
  have "?nsqn_le_nsqn dip" by simp (metis kD_nsqn_quality_increases_trans)
ultimately show ?thesis ..
qed
qed
} note update2 = this

have "opaadv i ⊨ (?S, ?U →) onl ΓAODV (λ(σ, _).
  ∀dip ∈ kD(rt (σ i)). the (nhop (rt (σ i) dip) ≠ dip
  → dip ∈ kD(rt (σ (the (nhop (rt (σ i) dip))))
  ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i) dip)))) dip))"
  by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
    oaadv_trans]
    onl_oinvariant_sterms [OF aadv_wf oaddpreRT_welldefined]
    onl_oinvariant_sterms [OF aadv_wf odests_vD_inc_sqn]
    onl_oinvariant_sterms [OF aadv_wf oprerr_guard]
    onl_oinvariant_sterms [OF aadv_wf rreq_sip]
    onl_oinvariant_sterms [OF aadv_wf rrep_sip]
    onl_oinvariant_sterms [OF aadv_wf rerr_sip]
    other_quality_increases
    other_localD
    solve: basic basic_prerr
    simp add: seqsimp nsqn_invalidate nhop_update_sip
    simp del: One_nat_def]
    (rule conjI, erule(2) update1, erule(2) update2)+

  thus ?thesis unfolding Let_def by auto
qed

```

Proposition 7.30

```

lemmas okD_unk_or_atleast_one =
  open_seq_invariant [OF kD_unk_or_atleast_one initiali_aadv,
    simplified seq1_onl_swap]

```

```

lemmas ozero_seq_unk_hops_one =
  open_seq_invariant [OF zero_seq_unk_hops_one initiali_aadv,
    simplified seq1_onl_swap]

```

lemma oreachable\_fresh\_okD\_unk\_or\_atleast\_one:

```

  fixes dip
  assumes "(σ, p) ∈ oreachable (opaadv i)
    (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
      ∧ msg_zhops m)))
    (other quality_increases {i})"
  and "dip∈kD(rt (σ i))"
  shows "π3(the (rt (σ i) dip)) = unk ∨ 1 ≤ π2(the (rt (σ i) dip))"
  (is "?P dip")
proof -
  have "∃l. l∈labels ΓAODV p" by (metis aadv_ex_label)
  with assms(1) have "∀dip∈kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaadv_trans aadv_trans]],
    auto dest!: otherwith_actionD onlD simp: seqsimp)
  with ⟨dip∈kD(rt (σ i))⟩ show ?thesis by simp
qed

```

lemma oreachable\_fresh\_ozero\_seq\_unk\_hops\_one:

```

  fixes dip
  assumes "(σ, p) ∈ oreachable (opaadv i)
    (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
      ∧ msg_zhops m)))
    (other quality_increases {i})"

```

```

    and "dip ∈ kD(rt (σ i))"
  shows "sqn (rt (σ i)) dip = 0 →
        sqnf (rt (σ i)) dip = unk
        ∧ the (dhops (rt (σ i)) dip) = 1
        ∧ the (nhop (rt (σ i)) dip) = dip"
  (is "?P dip")
proof -
  have "∃ l. l ∈ labels ΓAODV p" by (metis aodv_ex_label)
  with assms(1) have "∀ dip ∈ kD (rt (σ i)). ?P dip"
    by - (drule oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaodv_trans aodv_trans]],
          auto dest!: onlD otherwith_actionD simp: seqlsimp)
  with ⟨dip ∈ kD(rt (σ i))⟩ show ?thesis by simp
qed

lemma seq_nhop_quality_increases':
  shows "opaodv i ⊨ (otherwith (op=) {i}
        (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
        other quality_increases {i} →)
        onl ΓAODV (λ(σ, _). ∀dip. let nhop = the (nhop (rt (σ i)) dip)
        in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhop))
        ∧ nhop ≠ dip
        → (rt (σ i)) ⊑dip (rt (σ nhop)))"
  (is "_ ⊨ (?S i, _ →) _")
proof -
  have weaken:
    "∧p I Q R P. p ⊨ (otherwith quality_increases I (orecvmsg Q), other quality_increases I →) P
    ⇒ p ⊨ (otherwith (op=) I (orecvmsg (λσ m. Q σ m ∧ R σ m)), other quality_increases I →) P"
    by auto
  {
    fix i a and σ σ' :: "ip ⇒ state"
    assume a1: "∀dip. dip ∈ vD(rt (σ i))
              ∧ dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))
              ∧ (the (nhop (rt (σ i)) dip)) ≠ dip
              → rt (σ i) ⊑dip rt (σ (the (nhop (rt (σ i)) dip)))"
      and ow: "?S i σ σ' a"
    have "∀dip. dip ∈ vD(rt (σ i))
          ∧ dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))
          ∧ (the (nhop (rt (σ i)) dip)) ≠ dip
          → rt (σ i) ⊑dip rt (σ' (the (nhop (rt (σ i)) dip)))"
    proof clarify
      fix dip
      assume a2: "dip ∈ vD(rt (σ i))"
      and a3: "dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))"
      and a4: "(the (nhop (rt (σ i)) dip)) ≠ dip"
      from ow have "∀j. j ≠ i → σ j = σ' j" by auto
      show "rt (σ i) ⊑dip rt (σ' (the (nhop (rt (σ i)) dip)))"
      proof (cases "(the (nhop (rt (σ i)) dip)) = i")
        assume "(the (nhop (rt (σ i)) dip)) = i"
        with ⟨dip ∈ vD(rt (σ i))⟩ have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))" by simp
        with a1 a2 a4 have "rt (σ i) ⊑dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp
        with ⟨(the (nhop (rt (σ i)) dip)) = i⟩ have "rt (σ i) ⊑dip rt (σ i)" by simp
        hence False by simp
      thus ?thesis ..
    next
      assume "(the (nhop (rt (σ i)) dip)) ≠ i"
      with ⟨∀j. j ≠ i → σ j = σ' j⟩
      have *: "σ (the (nhop (rt (σ i)) dip)) = σ' (the (nhop (rt (σ i)) dip))" by simp
      with ⟨dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))⟩
      have "dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))" by simp
      with a1 a2 a4 have "rt (σ i) ⊑dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp
      with * show ?thesis by simp
    qed
  }
  qed
} note basic = this

```

```

{ fix  $\sigma$   $\sigma'$  a dip sip i
  assume a1: " $\forall$ dip. dip $\in$ vD(rt ( $\sigma$  i))
     $\wedge$  dip $\in$ vD(rt ( $\sigma$  (the (nhop (rt ( $\sigma$  i)) dip))))
     $\wedge$  the (nhop (rt ( $\sigma$  i)) dip)  $\neq$  dip
     $\rightarrow$  rt ( $\sigma$  i)  $\sqsubset_{\text{dip}}$  rt ( $\sigma$  (the (nhop (rt ( $\sigma$  i)) dip)))"
  and ow: "?S i  $\sigma$   $\sigma'$  a"
  have " $\forall$ dip. dip $\in$ vD(update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, {}))
     $\wedge$  dip $\in$ vD(rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, {})) dip))))
     $\wedge$  the (nhop (update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, {})) dip)  $\neq$  dip
     $\rightarrow$  update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, {})
       $\sqsubset_{\text{dip}}$  rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, {})) dip)))"
  proof clarify
    fix dip
    assume a2: "dip $\in$ vD (update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, {}))"
    and a3: "dip $\in$ vD(rt ( $\sigma'$  (the (nhop
      (update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, {})) dip))))"
    and a4: "the (nhop (update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, {})) dip)  $\neq$  dip"
    show "update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, {})
       $\sqsubset_{\text{dip}}$  rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, {})) dip)))"
    proof (cases "dip = sip")
      assume "dip = sip"
      with  $\langle$ the (nhop (update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, {})) dip)  $\neq$  dip $\rangle$ 
      have False by simp
      thus ?thesis ..
    next
      assume [simp]: "dip  $\neq$  sip"
      from a2 have "dip $\in$ vD(rt ( $\sigma$  i))  $\vee$  dip = sip"
        by (rule vD_update_val)
      with  $\langle$ dip  $\neq$  sip $\rangle$  have "dip $\in$ vD(rt ( $\sigma$  i))" by simp
      moreover from a3 have "dip $\in$ vD(rt ( $\sigma'$  (the (nhop (rt ( $\sigma$  i)) dip))))" by simp
      moreover from a4 have "the (nhop (rt ( $\sigma$  i)) dip)  $\neq$  dip" by simp
      ultimately have "rt ( $\sigma$  i)  $\sqsubset_{\text{dip}}$  rt ( $\sigma'$  (the (nhop (rt ( $\sigma$  i)) dip)))"
        using a1 ow by - (drule(1) basic, simp)
      with  $\langle$ dip  $\neq$  sip $\rangle$  show ?thesis
        by - (erule rt_strictly_fresher_update_other, simp)
    qed
  qed
} note update_0_unk = this

{ fix  $\sigma$  a  $\sigma'$  nhop
  assume pre: " $\forall$ dip. dip $\in$ vD(rt ( $\sigma$  i))  $\wedge$  dip $\in$ vD(rt ( $\sigma$  (nhop dip)))  $\wedge$  nhop dip  $\neq$  dip
     $\rightarrow$  rt ( $\sigma$  i)  $\sqsubset_{\text{dip}}$  rt ( $\sigma$  (nhop dip))"
  and ow: "?S i  $\sigma$   $\sigma'$  a"
  have " $\forall$ dip. dip  $\in$  vD (invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i)))
     $\wedge$  dip  $\in$  vD (rt ( $\sigma'$  (nhop dip)))  $\wedge$  nhop dip  $\neq$  dip
     $\rightarrow$  rt ( $\sigma$  i)  $\sqsubset_{\text{dip}}$  rt ( $\sigma'$  (nhop dip))"
  proof clarify
    fix dip
    assume "dip $\in$ vD(invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i)))"
      and "dip $\in$ vD(rt ( $\sigma'$  (nhop dip)))"
      and "nhop dip  $\neq$  dip"
    from this(1) have "dip $\in$ vD (rt ( $\sigma$  i))"
      by (clarsimp dest!: vD_invalidate_vD_not_dests)
    moreover from ow have " $\forall$ j. j  $\neq$  i  $\rightarrow$   $\sigma$  j =  $\sigma'$  j" by auto
    ultimately have "rt ( $\sigma$  i)  $\sqsubset_{\text{dip}}$  rt ( $\sigma$  (nhop dip))"
      using pre  $\langle$ dip  $\in$  vD (rt ( $\sigma'$  (nhop dip))) $\rangle$   $\langle$ nhop dip  $\neq$  dip $\rangle$ 
      by metis
    with  $\langle$  $\forall$ j. j  $\neq$  i  $\rightarrow$   $\sigma$  j =  $\sigma'$  j $\rangle$  show "rt ( $\sigma$  i)  $\sqsubset_{\text{dip}}$  rt ( $\sigma'$  (nhop dip))"
      by (metis rt_strictly_fresher_irefl)
    qed
  } note invalidate = this

{ fix  $\sigma$  a  $\sigma'$  dip oip osn sip hops i

```

```

assume pre: "∀dip. dip ∈ vD (rt (σ i))
            ∧ dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))
            ∧ the (nhop (rt (σ i)) dip) ≠ dip
            → rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))"
and ow: "?S i σ σ' a"
and "Suc 0 ≤ osn"
and a6: "sip ≠ oip → oip ∈ kD (rt (σ sip))
        ∧ osn ≤ nsqn (rt (σ sip)) oip
        ∧ (nsqn (rt (σ sip)) oip = osn
           → the (dhops (rt (σ sip)) oip) ≤ hops
              ∨ the (flag (rt (σ sip)) oip) = inv)"
and after: "σ' i = σ i (|rt := update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })|)"
have "∀dip. dip ∈ vD (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }))
    ∧ dip ∈ vD (rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, { })) dip))))
    ∧ the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })) dip) ≠ dip
    → update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })
    ⊆dip
    rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })) dip)))"
proof clarify
fix dip
assume a2: "dip ∈ vD (update (rt (σ i)) oip (osn, kno, val, Suc (hops), sip, { }))"
and a3: "dip ∈ vD (rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, { })) dip))))"
and a4: "the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })) dip) ≠ dip"
from ow have a5: "∀j. j ≠ i → σ j = σ' j" by auto
show "update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })
    ⊆dip
    rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })) dip)))"
(is "?rt1 ⊆dip ?rt2 dip")
proof (cases "?rt1 = rt (σ i)")
assume nochange [simp]:
"update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }) = rt (σ i)"

from after have "σ' i = σ i" by simp
with a5 have "∀j. σ j = σ' j" by metis

from a2 have "dip ∈ vD (rt (σ i))" by simp
moreover from a3 have "dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))"
using nochange and (∀j. σ j = σ' j) by clarsimp
moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))"
using pre by simp

hence "rt (σ i) ⊆dip rt (σ' (the (nhop (rt (σ i)) dip)))"
using (∀j. σ j = σ' j) by simp
thus "?thesis" by simp
next
assume change: "?rt1 ≠ rt (σ i)"
from after a2 have "dip ∈ kD (rt (σ' i))" by auto
show ?thesis
proof (cases "dip = oip")
assume "dip ≠ oip"

with a2 have "dip ∈ vD (rt (σ i))" by auto
moreover with a3 a5 after and (dip ≠ oip)
have "dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))"
by simp metis
moreover from a4 and (dip ≠ oip) have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))"
using pre by simp

with after and a5 and (dip ≠ oip) show ?thesis
by simp (metis rt_strictly_fresher_update_other

```



```

      rt_strictly_fresher_irefl)
next
  assume "dip = oip"

  with a4 and change have "sip ≠ oip" by simp
  with a6 have "oip ∈ kD(rt (σ sip))"
    and "osn ≤ nsqn (rt (σ sip)) oip" by auto

  from a3 change ⟨dip = oip⟩ have "oip ∈ vD(rt (σ' sip))" by simp
  hence "the (flag (rt (σ' sip)) oip) = val" by simp

  from ⟨oip ∈ kD(rt (σ sip))⟩
  have "osn < nsqn (rt (σ' sip)) oip ∨ (osn = nsqn (rt (σ' sip)) oip
    ∧ the (dhops (rt (σ' sip)) oip) ≤ hops)"
  proof
    assume "oip ∈ vD(rt (σ sip))"
    hence "the (flag (rt (σ sip)) oip) = val" by simp
    with a6 ⟨sip ≠ oip⟩ have "nsqn (rt (σ sip)) oip = osn →
      the (dhops (rt (σ sip)) oip) ≤ hops"
      by simp
    show ?thesis
  proof
    show ?thesis by auto
  next
    — alternative to using sip_not_ip
    assume [simp]: "sip = i"
    have "?rt1 = rt (σ i)"
    proof (rule update_cases_kD, simp_all)
      from ⟨Suc 0 ≤ osn⟩ show "0 < osn" by simp
    next
      from ⟨oip ∈ kD(rt (σ sip))⟩ and ⟨sip = i⟩ show "oip ∈ kD(rt (σ i))"
      by simp
    next
      assume "sqn (rt (σ i)) oip < osn"
      also from ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
      have "... ≤ nsqn (rt (σ i)) oip" by simp
      also have "... ≤ sqn (rt (σ i)) oip"
      by (rule nsqn_sqn)
      finally have "sqn (rt (σ i)) oip < sqn (rt (σ i)) oip" .
      hence False by simp
      thus "(λa. if a = oip
        then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
        else rt (σ i) a) = rt (σ i)" ..
    next
      assume "sqn (rt (σ i)) oip = osn"
      and "Suc hops < the (dhops (rt (σ i)) oip)"
      from this(1) and ⟨oip ∈ vD (rt (σ sip))⟩ have "nsqn (rt (σ i)) oip = osn"
      by simp
      with ⟨nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
      have "the (dhops (rt (σ i)) oip) ≤ hops" by simp
      with ⟨Suc hops < the (dhops (rt (σ i)) oip)⟩ have False by simp
      thus "(λa. if a = oip
        then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
        else rt (σ i) a) = rt (σ i)" ..
    next
      assume "the (flag (rt (σ i)) oip) = inv"
      with ⟨the (flag (rt (σ sip)) oip) = val⟩ have False by simp
      thus "(λa. if a = oip
        then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
        else rt (σ i) a) = rt (σ i)" ..
  end
end

```

```

next
  from ⟨oip∈kD(rt (σ sip))⟩
    show "(λa. if a = oip then Some (the (rt (σ i) oip)) else rt (σ i) a) = rt (σ i)"
      by (auto dest!: kD_Some)
qed
with change have False ..
thus ?thesis ..
qed
next
assume "oip∈iD(rt (σ sip))"
with ⟨the (flag (rt (σ' sip) oip) = val) and a5 have "sip = i"
  by (metis f.distinct(1) iD_flag_is_inv)
from ⟨oip∈iD(rt (σ sip))⟩ have "the (flag (rt (σ sip) oip) = inv" by auto
with ⟨sip = i⟩ ⟨Suc 0 ≤ osn⟩ change after ⟨oip∈kD(rt (σ sip))⟩
  have "nsqn (rt (σ sip) oip) < nsqn (rt (σ' sip) oip)"
    unfolding update_def
    by (clarsimp split: option.split_asm split_if_asm)
      (auto simp: sqn_def)
with ⟨osn ≤ nsqn (rt (σ sip) oip)⟩ have "osn < nsqn (rt (σ' sip) oip)"
  by simp
thus ?thesis ..
qed
thus ?thesis
proof
assume osnlt: "osn < nsqn (rt (σ' sip) oip)"
from ⟨dip∈kD(rt (σ' i))⟩ and ⟨dip = oip⟩ have "dip ∈ kD (?rt1)" by simp
moreover from a3 have "dip ∈ kD (?rt2 dip)" by simp
moreover have "nsqn ?rt1 dip < nsqn (?rt2 dip) dip"
  proof -
    have "nsqn ?rt1 oip = osn"
      by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
    also have "... < nsqn (rt (σ' sip) oip)" using osnlt .
    also have "... = nsqn (?rt2 oip) oip" by (simp add: change)
    finally show ?thesis
      using ⟨dip = oip⟩ by simp
  qed
ultimately show ?thesis
  by (rule rt_strictly_fresher_ltI)
next
assume osneq: "osn = nsqn (rt (σ' sip) oip) ∧ the (dhops (rt (σ' sip) oip) ≤ hops"

have "oip∈kD(?rt1)" by simp
moreover from a3 ⟨dip = oip⟩ have "oip∈kD(?rt2 oip)" by simp

moreover have "nsqn ?rt1 oip = nsqn (?rt2 oip) oip"
proof -
  from osneq have "osn = nsqn (rt (σ' sip) oip)" ..
  also have "osn = nsqn ?rt1 oip"
    by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
  also have "nsqn (rt (σ' sip) oip) = nsqn (?rt2 oip) oip"
    by (simp add: change)
  finally show ?thesis .
qed

moreover have "π5(the (?rt2 oip oip)) < π5(the (?rt1 oip))"
proof -
  from osneq have "the (dhops (rt (σ' sip) oip) ≤ hops" ..
  moreover from ⟨oip ∈ vD (rt (σ' sip))⟩ have "oip∈kD(rt (σ' sip))" by auto
  ultimately have "π5(the (rt (σ' sip) oip)) ≤ hops"
    by (auto simp add: proj5_eq_dhops)
  also from change after have "hops < π5(the (rt (σ' i) oip))"
    by (simp add: proj5_eq_dhops) (metis dhops_update_changed lessI)
  finally have "π5(the (rt (σ' sip) oip)) < π5(the (rt (σ' i) oip))" .
  with change after show ?thesis by simp

```

```

qed

ultimately have "?rt1  $\sqsubseteq_{oip}$  ?rt2 oip"
  by (rule rt_strictly_fresher_eqI)
  with ⟨dip = oip⟩ show ?thesis by simp
qed

qed
qed
qed
} note rreq_rrep_update = this

have "opaadv i  $\models$  (otherwith (op=) {i} (orecvmsg ( $\lambda\sigma m. \text{msg\_fresh } \sigma m$ 
   $\wedge \text{msg\_zhops } m$ )),
  other quality_increases {i}  $\rightarrow$ )
  onl  $\Gamma_{AODV}$ 
  ( $\lambda(\sigma, \_). \forall \text{dip}. \text{dip} \in \text{vD}(\text{rt } (\sigma \text{ i})) \cap \text{vD}(\text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}))))$ 
   $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}) \neq \text{dip}$ 
   $\rightarrow \text{rt } (\sigma \text{ i}) \sqsubseteq_{\text{dip}} \text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}))))$ ")
proof (inv_terms inv add: onl_oinvariant_sterms [OF aadv_wf rreq_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf rrep_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf rerr_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf oosn_rreq [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf odsn_rrep [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf oaddpreRT_welldefined])
  solve: basic update_0_unk invalidate rreq_rrep_update
  simp add: seqsimp)
fix  $\sigma \sigma' p l$ 
assume or: " $(\sigma, p) \in \text{oreachable } (\text{opaadv } i) (?S \text{ i})$  (other quality_increases {i})"
  and "other quality_increases {i}  $\sigma \sigma'$ "
  and ll: " $l \in \text{labels } \Gamma_{AODV} p$ "
  and pre: " $\forall \text{dip}. \text{dip} \in \text{vD}(\text{rt } (\sigma \text{ i}))$ 
   $\wedge \text{dip} \in \text{vD}(\text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}))))$ 
   $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}) \neq \text{dip}$ 
   $\rightarrow \text{rt } (\sigma \text{ i}) \sqsubseteq_{\text{dip}} \text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}))))$ "
from this(1-2)
  have or': " $(\sigma', p) \in \text{oreachable } (\text{opaadv } i) (?S \text{ i})$  (other quality_increases {i})"
  by - (rule oreachable_other')

from or and ll have next_hop: " $\forall \text{dip}. \text{let } \text{nhop} = \text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})$ 
  in  $\text{dip} \in \text{kD}(\text{rt } (\sigma \text{ i})) \wedge \text{nhop} \neq \text{dip}$ 
   $\rightarrow \text{dip} \in \text{kD}(\text{rt } (\sigma \text{ nhop}))$ 
   $\wedge \text{nsqn } (\text{rt } (\sigma \text{ i})) \text{ dip} \leq \text{nsqn } (\text{rt } (\sigma \text{ nhop})) \text{ dip}$ "
  by (auto dest!: onl_oinvariant_weakenD [OF seq_compare_next_hop'])

from or and ll have unk_hops_one: " $\forall \text{dip} \in \text{kD}(\text{rt } (\sigma \text{ i})). \text{sqn } (\text{rt } (\sigma \text{ i})) \text{ dip} = 0$ 
   $\rightarrow \text{sqnf } (\text{rt } (\sigma \text{ i})) \text{ dip} = \text{unk}$ 
   $\wedge \text{the } (\text{dhops } (\text{rt } (\sigma \text{ i})) \text{ dip}) = 1$ 
   $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}) = \text{dip}$ "
  by (auto dest!: onl_oinvariant_weakenD [OF ozero_seq_unk_hops_one
  [OF oaadv_trans aadv_trans]])
  otherwith_actionD
  simp: seqsimp)

from ⟨other quality_increases {i}  $\sigma \sigma'$ ⟩ have " $\sigma' \text{ i} = \sigma \text{ i}$ " by auto
hence "quality_increases ( $\sigma \text{ i}$ ) ( $\sigma' \text{ i}$ )" by auto
with ⟨other quality_increases {i}  $\sigma \sigma'$ ⟩ have " $\forall j. \text{quality\_increases } (\sigma \text{ j}) (\sigma' \text{ j})$ "
  by - (erule otherE, metis singleton_iff)

show " $\forall \text{dip}. \text{dip} \in \text{vD}(\text{rt } (\sigma' \text{ i}))$ 
   $\wedge \text{dip} \in \text{vD}(\text{rt } (\sigma' (\text{the } (\text{nhop } (\text{rt } (\sigma' \text{ i})) \text{ dip}))))$ 
   $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma' \text{ i})) \text{ dip}) \neq \text{dip}$ 
   $\rightarrow \text{rt } (\sigma' \text{ i}) \sqsubseteq_{\text{dip}} \text{rt } (\sigma' (\text{the } (\text{nhop } (\text{rt } (\sigma' \text{ i})) \text{ dip}))))$ "
proof clarify
  fix dip

```

```

assume "dip∈vD(rt (σ' i))"
  and "dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))"
  and "the (nhop (rt (σ' i)) dip) ≠ dip"
from this(1) and ⟨σ' i = σ i⟩ have "dip∈vD(rt (σ i))"
  and "dip∈kD(rt (σ i))"

by auto

from ⟨the (nhop (rt (σ' i)) dip) ≠ dip⟩ and ⟨σ' i = σ i⟩
  have "the (nhop (rt (σ i)) dip) ≠ dip" (is "?nhip ≠ _") by simp
with ⟨dip∈kD(rt (σ i))⟩ and next_hop
  have "dip∈kD(rt (σ (?nhip)))"
  and nsqns: "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ ?nhip)) dip"
  by (auto simp: Let_def)

have "0 < sqn (rt (σ i)) dip"
proof (rule neq0_conv [THEN iffD1, OF notI])
  assume "sqn (rt (σ i)) dip = 0"
  with ⟨dip∈kD(rt (σ i))⟩ and unk_hops_one
    have "?nhip = dip" by simp
  with ⟨?nhip ≠ dip⟩ show False ..
qed
also have "... = nsqn (rt (σ i)) dip"
  by (rule vD_nsqn_sqn [OF ⟨dip∈vD(rt (σ i))⟩, THEN sym])
also have "... ≤ nsqn (rt (σ ?nhip)) dip"
  by (rule nsqns)
also have "... ≤ sqn (rt (σ ?nhip)) dip"
  by (rule nsqn_sqn)
finally have "0 < sqn (rt (σ ?nhip)) dip" .

have "rt (σ i) ⊆dip rt (σ' ?nhip)"
proof (cases "dip∈vD(rt (σ ?nhip))")
  assume "dip∈vD(rt (σ ?nhip))"
  with pre ⟨dip∈vD(rt (σ i))⟩ and ⟨?nhip ≠ dip⟩
    have "rt (σ i) ⊆dip rt (σ ?nhip)" by auto
  moreover from ⟨∀j. quality_increases (σ j) (σ' j)⟩
    have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
  ultimately show ?thesis
    using ⟨dip∈kD(rt (σ ?nhip))⟩
    by (rule strictly_fresher_quality_increases_right)
next
  assume "dip∉vD(rt (σ ?nhip))"
  with ⟨dip∈kD(rt (σ ?nhip))⟩ have "dip∈iD(rt (σ ?nhip))" ..
  hence "the (flag (rt (σ ?nhip)) dip) = inv"
    by auto
  have "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ ?nhip)) dip"
    by (rule nsqns)
  also from ⟨dip∈iD(rt (σ ?nhip))⟩
    have "... = sqn (rt (σ ?nhip)) dip - 1" ..
  also have "... < sqn (rt (σ' ?nhip)) dip"
    proof -
      from ⟨∀j. quality_increases (σ j) (σ' j)⟩
        have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
      hence "∀ip. sqn (rt (σ ?nhip)) ip ≤ sqn (rt (σ' ?nhip)) ip" by auto
      hence "sqn (rt (σ ?nhip)) dip ≤ sqn (rt (σ' ?nhip)) dip" ..
      with ⟨0 < sqn (rt (σ ?nhip)) dip⟩ show ?thesis by auto
    qed
  also have "... = nsqn (rt (σ' ?nhip)) dip"
    proof (rule vD_nsqn_sqn [THEN sym])
      from ⟨dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and ⟨σ' i = σ i⟩
        show "dip∈vD(rt (σ' ?nhip))" by simp
    qed
  finally have "nsqn (rt (σ i)) dip < nsqn (rt (σ' ?nhip)) dip" .

moreover from ⟨dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and ⟨σ' i = σ i⟩

```

```

      have "dip ∈ kD(rt (σ' ?nhip))" by auto
      ultimately show "rt (σ i) ⊑dip rt (σ' ?nhip)"
      using ⟨dip ∈ kD(rt (σ i))⟩ by - (rule rt_strictly_fresher_ltI)
    qed
    with ⟨σ' i = σ i⟩ show "rt (σ' i) ⊑dip rt (σ' (the (nhop (rt (σ' i)) dip)))"
    by simp
  qed
  qed
  thus ?thesis unfolding Let_def .
end

```

```

lemma seq_compare_next_hop:
  fixes w
  shows "opaadv i ⊨ (otherwith (op=) {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
      in dip ∈ kD(rt (σ i)) ∧ nhip ≠ dip →
        dip ∈ kD(rt (σ nhip))
        ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip)"
  by (rule oinvariant_weakenE [OF seq_compare_next_hop']) (auto dest!: onID)

```

```

lemma seq_nhop_quality_increases:
  shows "opaadv i ⊨ (otherwith (op=) {i}
    (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} →)
    global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
      in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
        → (rt (σ i)) ⊑dip (rt (σ nhip)))"
  by (rule oinvariant_weakenE [OF seq_nhop_quality_increases']) (auto dest!: onID)

```

end

## 1.10 Routing graphs and loop freedom

```

theory A_Loop_Freedom
imports A_Aodv_Predicates A_Fresher
begin

```

Define the central theorem that relates an invariant over network states to the absence of loops in the associated routing graph.

**definition**

```

rt_graph :: "(ip ⇒ state) ⇒ ip ⇒ ip rel"
where
  "rt_graph σ = (λdip.
    {(ip, ip') | ip ip' dsn dsk hops pre.
      ip ≠ dip ∧ rt (σ ip) dip = Some (dsn, dsk, val, hops, ip', pre)})"

```

Given the state of a network  $\sigma$ , a routing graph for a given destination  $dip$  abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

```

lemma rt_graphE [elim]:
  fixes n dip ip ip'
  assumes "(ip, ip') ∈ rt_graph σ dip"
  shows "ip ≠ dip ∧ (∃r. rt (σ ip) = r
    ∧ (∃dsn dsk hops pre. r dip = Some (dsn, dsk, val, hops, ip', pre)))"
  using assms unfolding rt_graph_def by auto

```

```

lemma rt_graph_vD [dest]:
  "∧ip ip' σ dip. (ip, ip') ∈ rt_graph σ dip ⇒ dip ∈ vD(rt (σ ip))"
  unfolding rt_graph_def vD_def by auto

```

```

lemma rt_graph_vD_trans [dest]:
  "∧ip ip' σ dip. (ip, ip') ∈ (rt_graph σ dip)+ ⇒ dip ∈ vD(rt (σ ip))"
  by (erule converse_tranclE) auto

```

lemma rt\_graph\_not\_dip [dest]:

" $\bigwedge ip\ ip'\ \sigma\ dip.\ (ip, ip') \in rt\_graph\ \sigma\ dip \implies ip \neq dip$ "  
 unfolding rt\_graph\_def by auto

lemma rt\_graph\_not\_dip\_trans [dest]:

" $\bigwedge ip\ ip'\ \sigma\ dip.\ (ip, ip') \in (rt\_graph\ \sigma\ dip)^+ \implies ip \neq dip$ "  
 by (erule converse\_tranclE) auto

NB: the property below cannot be lifted to the transitive closure

lemma rt\_graph\_nhip\_is\_nhop [dest]:

" $\bigwedge ip\ ip'\ \sigma\ dip.\ (ip, ip') \in rt\_graph\ \sigma\ dip \implies ip' = the\ (nhop\ (rt\ (\sigma\ ip))\ dip)$ "  
 unfolding rt\_graph\_def by auto

theorem inv\_to\_loop\_freedom:

assumes " $\forall i\ dip.\ let\ nhip = the\ (nhop\ (rt\ (\sigma\ i))\ dip)$   
 in  $dip \in vD\ (rt\ (\sigma\ i)) \cap vD\ (rt\ (\sigma\ nhip)) \wedge nhip \neq dip$   
 $\longrightarrow (rt\ (\sigma\ i)) \sqsubset_{dip}\ (rt\ (\sigma\ nhip))$ "

shows " $\forall dip.\ irrefl\ ((rt\_graph\ \sigma\ dip)^+)$ "

using assms proof (intro allI)

fix  $\sigma :: "ip \Rightarrow state"$  and dip

assume inv: " $\forall ip\ dip.$

let  $nhip = the\ (nhop\ (rt\ (\sigma\ ip))\ dip)$   
 in  $dip \in vD\ (rt\ (\sigma\ ip)) \cap vD\ (rt\ (\sigma\ nhip)) \wedge$   
 $nhip \neq dip \longrightarrow rt\ (\sigma\ ip) \sqsubset_{dip}\ rt\ (\sigma\ nhip)$ "

{ fix ip ip'

assume " $(ip, ip') \in (rt\_graph\ \sigma\ dip)^+$ "

and " $dip \in vD\ (rt\ (\sigma\ ip'))$ "

and " $ip' \neq dip$ "

hence " $rt\ (\sigma\ ip) \sqsubset_{dip}\ rt\ (\sigma\ ip')$ "

proof induction

fix nhip

assume " $(ip, nhip) \in rt\_graph\ \sigma\ dip$ "

and " $dip \in vD\ (rt\ (\sigma\ nhip))$ "

and " $nhip \neq dip$ "

from  $\langle (ip, nhip) \in rt\_graph\ \sigma\ dip \rangle$  have "1:  $dip \in vD\ (rt\ (\sigma\ ip))$ "

and " $nhip = the\ (nhop\ (rt\ (\sigma\ ip))\ dip)$ "

by auto

from  $\langle dip \in vD\ (rt\ (\sigma\ ip)) \rangle$  and  $\langle dip \in vD\ (rt\ (\sigma\ nhip)) \rangle$

have " $dip \in vD\ (rt\ (\sigma\ ip)) \cap vD\ (rt\ (\sigma\ nhip))$ " ..

with  $\langle nhip = the\ (nhop\ (rt\ (\sigma\ ip))\ dip) \rangle$

and  $\langle nhip \neq dip \rangle$

and inv

show " $rt\ (\sigma\ ip) \sqsubset_{dip}\ rt\ (\sigma\ nhip)$ "

by (clarsimp simp: Let\_def)

next

fix nhip nhip'

assume " $(ip, nhip) \in (rt\_graph\ \sigma\ dip)^+$ "

and " $(nhip, nhip') \in rt\_graph\ \sigma\ dip$ "

and IH: " $\llbracket dip \in vD\ (rt\ (\sigma\ nhip)); nhip \neq dip \rrbracket \implies rt\ (\sigma\ ip) \sqsubset_{dip}\ rt\ (\sigma\ nhip)$ "

and " $dip \in vD\ (rt\ (\sigma\ nhip'))$ "

and " $nhip' \neq dip$ "

from  $\langle (nhip, nhip') \in rt\_graph\ \sigma\ dip \rangle$  have 1: " $dip \in vD\ (rt\ (\sigma\ nhip))$ "

and 2: " $nhip \neq dip$ "

and " $nhip' = the\ (nhop\ (rt\ (\sigma\ nhip))\ dip)$ "

by auto

from 1 2 have " $rt\ (\sigma\ ip) \sqsubset_{dip}\ rt\ (\sigma\ nhip)$ " by (rule IH)

also have " $rt\ (\sigma\ nhip) \sqsubset_{dip}\ rt\ (\sigma\ nhip')$ "

proof -

from  $\langle dip \in vD\ (rt\ (\sigma\ nhip)) \rangle$  and  $\langle dip \in vD\ (rt\ (\sigma\ nhip')) \rangle$

have " $dip \in vD\ (rt\ (\sigma\ nhip)) \cap vD\ (rt\ (\sigma\ nhip'))$ " ..

with  $\langle nhip' \neq dip \rangle$

and  $\langle nhip' = the\ (nhop\ (rt\ (\sigma\ nhip))\ dip) \rangle$

and inv

```

      show "rt (σ nhip) ⊆dip rt (σ nhip′)"
        by (clarsimp simp: Let_def)
    qed
  finally show "rt (σ ip) ⊆dip rt (σ nhip′)" .
qed } note fresher = this

```

```

show "irrefl ((rt_graph σ dip)+)"
unfolding irrefl_def proof (intro allI notI)
  fix ip
  assume "(ip, ip) ∈ (rt_graph σ dip)+"
  moreover then have "dip ∈ vD(rt (σ ip))"
    and "ip ≠ dip"
    by auto
  ultimately have "rt (σ ip) ⊆dip rt (σ ip)" by (rule fresher)
  thus False by simp
qed
qed

```

end

## 1.11 Lift and transfer invariants to show loop freedom

```

theory A_Aodv_Loop_Freedom
imports "../.../AWN/OClosed_Transfer" "../.../AWN/Qmsg_Lifting" A_Global_Invariants A_Loop_Freedom
begin

```

lift to parallel processes with queues

```

lemma par_step_no_change_on_send_or_receive:
  fixes σ s a σ' s'
  assumes "((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos ΓAODV i) (seqp_sos ΓQMSG)"
    and "a ≠ τ"
  shows "σ' i = σ i"
  using assms by (rule qmsg_no_change_on_send_or_receive)

```

lemma par\_nhop\_quality\_increases:

```

shows "opaodv i << i qmsg ⊢ (otherwith (op=) {i} (orecvmsg (λσ m.
  msg_fresh σ m ∧ msg_zhops m)),
  other quality_increases {i} →)
  global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊆dip (rt (σ nhip)))"

```

```

proof (rule lift_into_qmsg [OF seq_nhop_quality_increases])

```

```

show "opaodv i ⊢A (otherwith (op=) {i}
  (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
  other quality_increases {i} →)
  globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"

```

```

proof (rule ostep_invariant_weakenE [OF oquality_increases], simp_all)

```

```

  fix t :: "((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"

```

```

  assume "onll ΓAODV (λ((σ, _), _, (σ', _)). ∀j. quality_increases (σ j) (σ' j)) t"

```

```

  thus "quality_increases (fst (fst t) i) (fst (snd (snd t)) i)"

```

```

    by (cases t) (clarsimp dest!: onllD, metis aodv_ex_label)

```

```

next

```

```

  fix σ σ' a

```

```

  assume "otherwith (op=) {i}
    (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)) σ σ' a"

```

```

  thus "otherwith quality_increases {i} (orecvmsg (λ_. rreq_rrep_sn)) σ σ' a"
    by - (erule weaken_otherwith, auto)

```

```

qed

```

```

qed auto

```

lemma par\_rreq\_rrep\_sn\_quality\_increases:

```

"opaodv i << i qmsg ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
  globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"

```

```

proof -

```

```

have "opaadv i  $\models_A (\lambda\sigma \_ . \text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\sigma, \_ , \sigma') . \text{quality\_increases } (\sigma \ i) (\sigma' \ i))$ "
  by (rule ostep_invariant_weakenE [OF olocal_quality_increases])
      (auto dest!: onllD seqllD elim!: aadv_ex_labelE)
hence "opaadv i  $\langle\langle_i \text{qmsg } \models_A (\lambda\sigma \_ . \text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\sigma, \_ , \sigma') . \text{quality\_increases } (\sigma \ i) (\sigma' \ i))$ "
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

```

lemma par\_rreq\_rrep\_nsqn\_fresh\_any\_step:

```

shows "opaadv i  $\langle\langle_i \text{qmsg } \models_A (\lambda\sigma \_ . \text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn}) \sigma,$ 
      other  $(\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\sigma, a, \sigma') . \text{anycast } (\text{msg\_fresh } \sigma) a)$ "

```

proof -

```

have "opaadv i  $\models_A (\lambda\sigma \_ . (\text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn})) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\sigma, a, \sigma') . \text{anycast } (\text{msg\_fresh } \sigma) a)$ "

```

```

proof (rule ostep_invariant_weakenE [OF rreq_rrep_nsqn_fresh_any_step_invariant])

```

fix t

```

assume "onll  $\Gamma_{AODV} (\lambda((\sigma, \_), a, \_). \text{anycast } (\text{msg\_fresh } \sigma) a) t$ "

```

```

thus "globala  $(\lambda(\sigma, a, \sigma') . \text{anycast } (\text{msg\_fresh } \sigma) a) t$ "

```

```

  by (cases t) (clarsimp dest!: onllD, metis aadv_ex_label)

```

qed auto

```

hence "opaadv i  $\langle\langle_i \text{qmsg } \models_A (\lambda\sigma \_ . (\text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn})) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\sigma, a, \sigma') . \text{anycast } (\text{msg\_fresh } \sigma) a)$ "

```

```

  by (rule lift_step_into_qmsg_statelessassm) simp_all

```

```

thus ?thesis by rule auto

```

qed

lemma par\_anycast\_msg\_zhops:

```

shows "opaadv i  $\langle\langle_i \text{qmsg } \models_A (\lambda\sigma \_ . \text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a)$ "

```

proof -

```

from anycast_msg_zhops initiali_aadv oaadv_trans aadv_trans

```

```

have "opaadv i  $\models_A (\text{act } TT, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      seqll i (onll  $\Gamma_{AODV} (\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a))$ "

```

```

  by (rule open_seq_step_invariant)

```

```

hence "opaadv i  $\models_A (\lambda\sigma \_ . \text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a)$ "

```

```

proof (rule ostep_invariant_weakenE)

```

```

fix t :: " $((\text{nat} \Rightarrow \text{state}) \times (\text{state}, \text{msg}, \text{pseqp}, \text{pseqp label}) \text{seqp}), \text{msg seq\_action})$  transition"

```

```

assume "seqll i (onll  $\Gamma_{AODV} (\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a)) t$ "

```

```

thus "globala  $(\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a) t$ "

```

```

  by (cases t) (clarsimp dest!: seqllD onllD, metis aadv_ex_label)

```

qed simp\_all

```

hence "opaadv i  $\langle\langle_i \text{qmsg } \models_A (\lambda\sigma \_ . \text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a)$ "

```

```

  by (rule lift_step_into_qmsg_statelessassm) simp_all

```

```

thus ?thesis by rule auto

```

qed

### 1.11.1 Lift to nodes

lemma node\_step\_no\_change\_on\_send\_or\_receive:

```

assumes " $((\sigma, \text{NodeS } i \ P \ R), a, (\sigma', \text{NodeS } i' \ P' \ R')) \in \text{onode\_sos}$ 
      (oparp_sos i (oseqp_sos  $\Gamma_{AODV}$  i) (seqp_sos  $\Gamma_{QMSG}$ ))"

```

```

  and "a  $\neq \tau$ "

```

```

  shows " $\sigma' \ i = \sigma \ i$ "

```

using assms

```

by (cases a) (auto elim!: par_step_no_change_on_send_or_receive)

```

lemma node\_nhop\_quality\_increases:

```

shows " $\langle i : \text{opaadv } i \langle\langle_i \text{qmsg} : R \rangle_o \models$ 
      (otherwith (op=)  $\{i\}$ )"

```



```

      (oarrivemsg ( $\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m$ )),
      other quality_increases {i}
    → global ( $\lambda\sigma. \forall \text{dip. let nhop} = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip})$ 
      in  $\text{dip} \in \text{vD } (\text{rt } (\sigma i)) \cap \text{vD } (\text{rt } (\sigma \text{nhop})) \wedge \text{nhop} \neq \text{dip}$ 
      →  $(\text{rt } (\sigma i)) \sqsubseteq_{\text{dip}} (\text{rt } (\sigma \text{nhop}))$ )"
  by (rule node_lift [OF par_nhop_quality_increases]) auto

```

lemma node\_quality\_increases:

```

"⟨ i : opaadv i ⟨⟨i qmsg : R ⟩o ⟩A (λσ -. oarrivemsg (λ_. rreq_rrep_sn) σ,
  other (λ_ -. True) {i} →)
  globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
  by (rule node_lift_step_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp

```

lemma node\_rreq\_rrep\_nsqn\_fresh\_any\_step:

```

shows "⟨ i : opaadv i ⟨⟨i qmsg : R ⟩o ⟩A
  (λσ -. oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ -. True) {i} →)
  globala (λ(σ, a, σ'). castmsg (msg_fresh σ) a)"
  by (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_nsqn_fresh_any_step])

```

lemma node\_anycast\_msg\_zhops:

```

shows "⟨ i : opaadv i ⟨⟨i qmsg : R ⟩o ⟩A
  (λσ -. oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ -. True) {i} →)
  globala (λ(_, a, _). castmsg msg_zhops a)"
  by (rule node_lift_anycast_statelessassm [OF par_anycast_msg_zhops])

```

lemma node\_silent\_change\_only:

```

shows "⟨ i : opaadv i ⟨⟨i qmsg : Ri ⟩o ⟩A (λσ -. oarrivemsg (λ_ -. True) σ,
  other (λ_ -. True) {i} →)
  globala (λ(σ, a, σ'). a ≠ τ → σ' i = σ i)"

```

proof (rule ostep\_invariantI, simp (no\_asm), rule impI)

```

  fix σ ζ a σ' ζ'
  assume or: "(σ, ζ) ∈ oreachable (⟨⟨i qmsg : Ri ⟩o ⟩
    (λσ -. oarrivemsg (λ_ -. True) σ)
    (other (λ_ -. True) {i}))"
  and tr: "((σ, ζ), a, (σ', ζ')) ∈ trans (⟨⟨i qmsg : Ri ⟩o ⟩)"
  and "a ≠ τn"
  from or obtain p R where "ζ = NodeS i p R"
  by - (drule node_net_state, metis)
  with tr have "((σ, NodeS i p R), a, (σ', ζ'))
    ∈ onode_sos (oparp_sos i (trans (opaadv i)) (trans qmsg))"
  by simp
  thus "σ' i = σ i" using ⟨a ≠ τn⟩
  by (cases rule: onode_sos.cases)
    (auto elim: qmsg_no_change_on_send_or_receive)

```

qed

### 1.11.2 Lift to partial networks

lemma arrive\_rreq\_rrep\_nsqn\_fresh\_inc\_sn [simp]:

```

assumes "oarrivemsg (λσ m. msg_fresh σ m ∧ P σ m) σ m"
shows "oarrivemsg (λ_. rreq_rrep_sn) σ m"
using assms by (cases m) auto

```

lemma opnet\_nhop\_quality\_increases:

```

shows "opnet (λi. opaadv i ⟨⟨i qmsg ⟩ p ⟩
  (otherwith (op=) (net_tree_ips p)
    (oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
  other quality_increases (net_tree_ips p) →)
  global (λσ. ∀ i ∈ net_tree_ips p. ∀ dip.
    let nhop = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhop)) ∧ nhop ≠ dip
    → (rt (σ i)) ⊑dip (rt (σ nhop)))"

```

proof (rule pnet\_lift [OF node\_nhop\_quality\_increases])

fix i R

```

have "{i : opaadv i <<_i qmsg : R>_o} ⊨A (λσ _ . oarrivemsg (λ_ . rreq_rrep_sn) σ,
      other (λ_ _ . True) {i} →) globala (λ(σ, a, σ').
      castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
proof (rule ostep_invariantI, simp (no_asm))
  fix σ s a σ' s'
  assume or: "(σ, s) ∈ oreachable (<<_i qmsg : R>_o)
      (λσ _ . oarrivemsg (λ_ . rreq_rrep_sn) σ)
      (other (λ_ _ . True) {i})"
  and tr: "((σ, s), a, (σ', s')) ∈ trans (<<_i qmsg : R>_o)"
  and am: "oarrivemsg (λ_ . rreq_rrep_sn) σ a"
  from or tr am have "castmsg (msg_fresh σ) a"
  by (auto dest!: ostep_invariantD [OF node_rreq_rrep_nsqn_fresh_any_step])
  moreover from or tr am have "castmsg (msg_zhops) a"
  by (auto dest!: ostep_invariantD [OF node_anycast_msg_zhops])
  ultimately show "castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a"
  by (case_tac a) auto
qed
thus "{i : opaadv i <<_i qmsg : R>_o} ⊨A
  (λσ _ . oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
  other quality_increases {i} →) globala (λ(σ, a, _).
  castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
  by rule auto
next
fix i R
show "{i : opaadv i <<_i qmsg : R>_o} ⊨A
  (λσ _ . oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
  other quality_increases {i} →) globala (λ(σ, a, σ').
  a ≠ τ ∧ (∀i d. a = i:deliver(d)) → σ i = σ' i)"
  by (rule ostep_invariant_weakenE [OF node_silent_change_only]) auto
next
fix i R
show "{i : opaadv i <<_i qmsg : R>_o} ⊨A
  (λσ _ . oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
  other quality_increases {i} →) globala (λ(σ, a, σ').
  a = τ ∨ (∃d. a = i:deliver(d)) → quality_increases (σ i) (σ' i))"
  by (rule ostep_invariant_weakenE [OF node_quality_increases]) auto
qed simp_all

```

### 1.11.3 Lift to closed networks

lemma onet\_nhop\_quality\_increases:

```

shows "oclosed (opnet (λi. opaadv i <<_i qmsg> p)
  ⊨ (λ_ _ _ . True, other quality_increases (net_tree_ips p) →)
  global (λσ. ∀i∈net_tree_ips p. ∀dip.
    let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊑dip (rt (σ nhip))))"

```

(is "\_ ⊨ (\_, ?U →) ?inv")

proof (rule inclosed\_closed)

from opnet\_nhop\_quality\_increases

```

show "opnet (λi. opaadv i <<_i qmsg> p)
  ⊨ (otherwith (op=) (net_tree_ips p) inclosed, ?U →) ?inv"

```

proof (rule oinvariant\_weakenE)

```

fix σ σ' :: "ip ⇒ state" and a :: "msg node_action"
assume "otherwith (op=) (net_tree_ips p) inclosed σ σ' a"
thus "otherwith (op=) (net_tree_ips p)
  (oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m)) σ σ' a"

```

proof (rule otherwithEI)

```

fix σ :: "ip ⇒ state" and a :: "msg node_action"
assume "inclosed σ a"
thus "oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ a"
proof (cases a)
  fix ii ni ms
  assume "a = ii-ni:arrive(ms)"

```

```

    moreover with ⟨inoclosed σ a⟩ obtain d di where "ms = newpkt(d, di)"
    by (cases ms) auto
    ultimately show ?thesis by simp
  qed simp_all
  qed
  qed
  qed

```

#### 1.11.4 Transfer into the standard model

```

interpretation aadv_openproc: openproc paadv opaadv id
  where "aadv_openproc.initmissing = initmissing"
  proof -
    show "openproc paadv opaadv id"
    proof unfold_locales
      fix i :: ip
      have "{(σ, ζ). (σ i, ζ) ∈ σAADV i ∧ (∀j. j ≠ i → σ j ∈ fst ' σAADV j)} ⊆ σAADV'"
        unfolding σAADV_def σAADV'_def
        proof (rule equalityD1)
          show "∧f p. {(σ, ζ). (σ i, ζ) ∈ {(f i, p)} ∧ (∀j. j ≠ i
            → σ j ∈ fst ' {(f j, p)}')} = {(f, p)}"
            by (rule set_eqI) auto
        qed
      thus "{(σ, ζ) | σ ζ s. s ∈ init (paadv i)
        ∧ (σ i, ζ) = id s
        ∧ (∀j. j ≠ i → σ j ∈ (fst o id) ' init (paadv j)) } ⊆ init (opaadv i)"
        by simp
    next
      show "∀j. init (paadv j) ≠ {}"
        unfolding σAADV_def by simp
    next
      fix i s a s' σ σ'
      assume "σ i = fst (id s)"
        and "σ' i = fst (id s')"
        and "(s, a, s') ∈ trans (paadv i)"
      then obtain q q' where "s = (σ i, q)"
        and "s' = (σ' i, q')"
        and "((σ i, q), a, (σ' i, q')) ∈ trans (paadv i)"
        by (cases s, cases s') auto
      from this(3) have "((σ, q), a, (σ', q')) ∈ trans (opaadv i)"
        by simp (rule open_seqp_action [OF aadv_wf])
      with ⟨s = (σ i, q)⟩ and ⟨s' = (σ' i, q')⟩
        show "((σ, snd (id s)), a, (σ', snd (id s')))) ∈ trans (opaadv i)"
        by simp
    qed
    then interpret op: openproc paadv opaadv id .
    have [simp]: "∧i. (SOME x. x ∈ (fst o id) ' init (paadv i)) = aadv_init i"
      unfolding σAADV_def by simp
    hence "∧i. openproc.initmissing paadv id i = initmissing i"
      unfolding op.initmissing_def op.someinit_def initmissing_def
      by (auto split: option.split)
    thus "openproc.initmissing paadv id = initmissing" ..
  qed

interpretation aadv_openproc_par_qmsg: openproc_parq paadv opaadv id qmsg
  where "aadv_openproc_par_qmsg.netglobal = netglobal"
    and "aadv_openproc_par_qmsg.initmissing = initmissing"
  proof -
    show "openproc_parq paadv opaadv id qmsg"
      by (unfold_locales) simp
    then interpret opq: openproc_parq paadv opaadv id qmsg .

    have im: "∧σ. openproc.initmissing (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) σ

```

```

                                                                    = initmissing  $\sigma$ "
  unfolding opq.initmissing_def opq.someinit_def initmissing_def
  unfolding  $\sigma_{AODV\_def}$   $\sigma_{QMSG\_def}$  by (clarsimp cong: option.case_cong)
  thus "openproc.initmissing ( $\lambda i$ . paadv i  $\langle\langle$  qmsg) ( $\lambda(p, q)$ . (fst (id p), snd (id p), q)) = initmissing"
  by (rule ext)
  have " $\bigwedge P \sigma$ . openproc.netglobal ( $\lambda i$ . paadv i  $\langle\langle$  qmsg) ( $\lambda(p, q)$ . (fst (id p), snd (id p), q)) P  $\sigma$ 
                                                                    = netglobal P  $\sigma$ "
  unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def
  unfolding  $\sigma_{AODV\_def}$   $\sigma_{QMSG\_def}$ 
  by (clarsimp cong: option.case_cong
      simp del: One_nat_def
      simp add: fst_initmissing_netgmap_default_aadv_init_netlift
              [symmetric, unfolded initmissing_def])
  thus "openproc.netglobal ( $\lambda i$ . paadv i  $\langle\langle$  qmsg) ( $\lambda(p, q)$ . (fst (id p), snd (id p), q)) = netglobal"
  by auto
qed

```

lemma net\_nhop\_quality\_increases:

```

  assumes "wf_net_tree n"
  shows "closed (pnet ( $\lambda i$ . paadv i  $\langle\langle$  qmsg) n)  $\models$  netglobal
        ( $\lambda\sigma$ .  $\forall i$  dip. let nhop = the (nhop (rt ( $\sigma$  i)) dip)
                          in dip  $\in$  vD (rt ( $\sigma$  i))  $\cap$  vD (rt ( $\sigma$  nhop))  $\wedge$  nhop  $\neq$  dip
                           $\longrightarrow$  (rt ( $\sigma$  i))  $\sqsubset_{dip}$  (rt ( $\sigma$  nhop)))"
  (is "_  $\models$  netglobal ( $\lambda\sigma$ .  $\forall i$ . ?inv  $\sigma$  i)")

```

proof -

```

  from (wf_net_tree n)
  have proto: "closed (pnet ( $\lambda i$ . paadv i  $\langle\langle$  qmsg) n)  $\models$  netglobal ( $\lambda\sigma$ .  $\forall i \in$  net_tree_ips n.  $\forall$  dip.
    let nhop = the (nhop (rt ( $\sigma$  i)) dip)
    in dip  $\in$  vD (rt ( $\sigma$  i))  $\cap$  vD (rt ( $\sigma$  nhop))  $\wedge$  nhop  $\neq$  dip
     $\longrightarrow$  (rt ( $\sigma$  i))  $\sqsubset_{dip}$  (rt ( $\sigma$  nhop)))"

```

by (rule aadv\_openproc\_par\_qmsg.close\_opnet [OF \_ onet\_nhop\_quality\_increases])

show ?thesis

unfolding invariant\_def opnet\_sos.opnet\_tau1

```

proof (rule, simp only: aadv_openproc_par_qmsg.netglobalsimp
                      fst_initmissing_netgmap_pair_fst, rule allI)

```

fix  $\sigma$  i

assume sr: " $\sigma \in$  reachable (closed (pnet ( $\lambda i$ . paadv i  $\langle\langle$  qmsg) n) TT)"

hence " $\forall i \in$  net\_tree\_ips n. ?inv (fst (initmissing (netgmap fst  $\sigma$ ))) i"

```

  by - (drule invariantD [OF proto],
        simp only: aadv_openproc_par_qmsg.netglobalsimp
                  fst_initmissing_netgmap_pair_fst)

```

thus "?inv (fst (initmissing (netgmap fst  $\sigma$ ))) i"

proof (cases "i  $\in$  net\_tree\_ips n")

assume "i  $\notin$  net\_tree\_ips n"

from sr have " $\sigma \in$  reachable (pnet ( $\lambda i$ . paadv i  $\langle\langle$  qmsg) n) TT" ..

hence "net\_ips  $\sigma$  = net\_tree\_ips n" ..

with (i  $\notin$  net\_tree\_ips n) have "i  $\notin$  net\_ips  $\sigma$ " by simp

hence "(fst (initmissing (netgmap fst  $\sigma$ ))) i = aadv\_init i"

by simp

thus ?thesis by simp

qed metis

qed

qed

### 1.11.5 Loop freedom of AODV

theorem aadv\_loop\_freedom:

assumes "wf\_net\_tree n"

shows "closed (pnet ( $\lambda i$ . paadv i  $\langle\langle$  qmsg) n)  $\models$  netglobal ( $\lambda\sigma$ .  $\forall$  dip. irrefl ((rt\_graph  $\sigma$  dip)<sup>+</sup>))"

using assms by (rule aadv\_openproc\_par\_qmsg.netglobal\_weakenE

[OF net\_nhop\_quality\_increases inv\_to\_loop\_freedom])

end

# Chapter 2

## Variant B: Forwarding the Route Reply

Explanation [4, §10.2]: In AODV's route discovery process, a RREP message from the destination node is unicast back along a route towards the originator of the RREQ message. Every intermediate node on the selected route will process the RREP message and, in most cases, forward it towards the originator node. However, there is a possibility that the RREP message is discarded at an intermediate node, which results in the originator node not receiving a reply. The discarding of the RREP message is due to the RFC specification of AODV [6] stating that an intermediate node only forwards the RREP message if it is not the originator node and it has created or updated a routing table entry to the destination node described in the RREP message. The latter requirement means that if a valid routing table entry to the destination node already exists, and is not updated when processing the RREP message, then the intermediate node will not forward the message. A solution to this problem is to require intermediate nodes to forward all RREP messages that they receive.

### 2.1 Predicates and functions used in the AODV model

```
theory B_Aodv_Data
imports B_FwdrrEPS
begin
```

#### 2.1.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

```
definition inc :: "sqn  $\Rightarrow$  sqn"
  where "inc sn  $\equiv$  if sn = 0 then sn else sn + 1"
```

```
lemma less_than_inc [simp]: "x  $\leq$  inc x"
  unfolding inc_def by simp
```

```
lemma inc_minus_suc_0 [simp]:
  "inc x - Suc 0 = x"
  unfolding inc_def by simp
```

```
lemma inc_never_one' [simp, intro]: "inc x  $\neq$  Suc 0"
  unfolding inc_def by simp
```

```
lemma inc_never_one [simp, intro]: "inc x  $\neq$  1"
  by simp
```

#### 2.1.2 Modelling Routes

A route is a 6-tuple,  $(dsn, dsk, flag, hops, nhop, pre)$  where  $dsn$  is the 'destination sequence number',  $dsk$  is the 'destination-sequence-number status',  $flag$  is the route status,  $hops$  is the number of hops to the destination,  $nhop$  is the next hop toward the destination, and  $pre$  is the set of 'precursor nodes' those interested in hearing about changes to the route.

```
type_synonym r = "sqn  $\times$  k  $\times$  f  $\times$  nat  $\times$  ip  $\times$  ip set"
```

```
definition proj2 :: "r  $\Rightarrow$  sqn" (" $\pi_2$ ")
```

```

where "π2 ≡ λ(dsn, _, _, _, _). dsn"

definition proj3 :: "r ⇒ k" ("π3")
  where "π3 ≡ λ(_, dsk, _, _, _). dsk"

definition proj4 :: "r ⇒ f" ("π4")
  where "π4 ≡ λ(_, _, flag, _, _). flag"

definition proj5 :: "r ⇒ nat" ("π5")
  where "π5 ≡ λ(_, _, _, hops, _, _). hops"

definition proj6 :: "r ⇒ ip" ("π6")
  where "π6 ≡ λ(_, _, _, _, nhip, _). nhip"

definition proj7 :: "r ⇒ ip set" ("π7")
  where "π7 ≡ λ(_, _, _, _, _, pre). pre"

lemma projs [simp]:
  "π2(dsn, dsk, flag, hops, nhip, pre) = dsn"
  "π3(dsn, dsk, flag, hops, nhip, pre) = dsk"
  "π4(dsn, dsk, flag, hops, nhip, pre) = flag"
  "π5(dsn, dsk, flag, hops, nhip, pre) = hops"
  "π6(dsn, dsk, flag, hops, nhip, pre) = nhip"
  "π7(dsn, dsk, flag, hops, nhip, pre) = pre"
  by (clarsimp simp: proj2_def proj3_def proj4_def
      proj5_def proj6_def proj7_def)+

lemma proj3_pred [intro]: "[[ P kno; P unk ] ⇒ P (π3 x)]"
  by (rule k.induct)

lemma proj4_pred [intro]: "[[ P val; P inv ] ⇒ P (π4 x)]"
  by (rule f.induct)

lemma proj6_pair_snd [simp]:
  fixes dsn' r
  shows "π6 (dsn', snd (r)) = π6(r)"
  by (cases r) simp



### 2.1.3 Routing Tables



Routing tables map ip addresses to route entries.



```

type_synonym rt = "ip ↦ r"

syntax
  "_Sigma_route" :: "rt ⇒ ip ↦ r" ("σroute'(_, _)'")

translations
  "σroute(rt, dip)" => "rt dip"

definition sqn :: "rt ⇒ ip ⇒ sqn"
  where "sqn rt dip ≡ case σroute(rt, dip) of Some r ⇒ π2(r) | None ⇒ 0"

definition sqnf :: "rt ⇒ ip ⇒ k"
  where "sqnf rt dip ≡ case σroute(rt, dip) of Some r ⇒ π3(r) | None ⇒ unk"

abbreviation flag :: "rt ⇒ ip ↦ f"
  where "flag rt dip ≡ map_option π4 (σroute(rt, dip))"

abbreviation dhops :: "rt ⇒ ip ↦ nat"
  where "dhops rt dip ≡ map_option π5 (σroute(rt, dip))"

abbreviation nhop :: "rt ⇒ ip ↦ ip"
  where "nhop rt dip ≡ map_option π6 (σroute(rt, dip))"

```


```

```

abbreviation precS :: "rt  $\Rightarrow$  ip  $\rightarrow$  ip set"
  where "precS rt dip  $\equiv$  map_option  $\pi_7$  ( $\sigma_{route}(rt, dip)$ )"

definition vD :: "rt  $\Rightarrow$  ip set"
  where "vD rt  $\equiv$  {dip. flag rt dip = Some val}"

definition iD :: "rt  $\Rightarrow$  ip set"
  where "iD rt  $\equiv$  {dip. flag rt dip = Some inv}"

definition kD :: "rt  $\Rightarrow$  ip set"
  where "kD rt  $\equiv$  {dip. rt dip  $\neq$  None}"

lemma kD_is_vD_and_iD: "kD rt = vD rt  $\cup$  iD rt"
  unfolding kD_def vD_def iD_def by auto

lemma vD_iD_gives_kD [simp]:
  " $\bigwedge$  ip rt. ip  $\in$  vD rt  $\implies$  ip  $\in$  kD rt"
  " $\bigwedge$  ip rt. ip  $\in$  iD rt  $\implies$  ip  $\in$  kD rt"
  unfolding kD_is_vD_and_iD by simp_all

lemma kD_Some [dest]:
  fixes dip rt
  assumes "dip  $\in$  kD rt"
  shows " $\exists$  dsn dsk flag hops nhip pre.
     $\sigma_{route}(rt, dip) = \text{Some } (dsn, dsk, flag, hops, nhip, pre)$ "
  using assms unfolding kD_def by simp

lemma kD_None [dest]:
  fixes dip rt
  assumes "dip  $\notin$  kD rt"
  shows " $\sigma_{route}(rt, dip) = \text{None}$ "
  using assms unfolding kD_def
  by (metis (mono_tags) mem_Collect_eq)

lemma vD_Some [dest]:
  fixes dip rt
  assumes "dip  $\in$  vD rt"
  shows " $\exists$  dsn dsk hops nhip pre.
     $\sigma_{route}(rt, dip) = \text{Some } (dsn, dsk, val, hops, nhip, pre)$ "
  using assms unfolding vD_def by simp

lemma vD_empty [simp]: "vD Map.empty = {}"
  unfolding vD_def by simp

lemma iD_Some [dest]:
  fixes dip rt
  assumes "dip  $\in$  iD rt"
  shows " $\exists$  dsn dsk hops nhip pre.
     $\sigma_{route}(rt, dip) = \text{Some } (dsn, dsk, inv, hops, nhip, pre)$ "
  using assms unfolding iD_def by simp

lemma val_is_vD [elim]:
  fixes ip rt
  assumes "ip  $\in$  kD(rt)"
  and "the (flag rt ip) = val"
  shows "ip  $\in$  vD(rt)"
  using assms unfolding vD_def by auto

lemma inv_is_iD [elim]:
  fixes ip rt
  assumes "ip  $\in$  kD(rt)"
  and "the (flag rt ip) = inv"
  shows "ip  $\in$  iD(rt)"
  using assms unfolding iD_def by auto

```

```

lemma iD_flag_is_inv [elim, simp]:
  fixes ip rt
  assumes "ip ∈ iD(rt)"
  shows "the (flag rt ip) = inv"
proof -
  from ⟨ip ∈ iD(rt)⟩ have "ip ∈ kD(rt)" by auto
  with assms show ?thesis unfolding iD_def by auto
qed

lemma kD_but_not_vD_is_iD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "ip ∉ vD(rt)"
  shows "ip ∈ iD(rt)"
proof -
  from ⟨ip ∈ kD(rt)⟩ obtain dsn dsk f hops nhop pre
  where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop, pre)"
  by (metis kD_Some)
  from ⟨ip ∉ vD(rt)⟩ have "f ≠ val"
  proof (rule contrapos_nn)
    assume "f = val"
    with rtip have "the (flag rt ip) = val" by simp
    with ⟨ip ∈ kD(rt)⟩ show "ip ∈ vD(rt)" ..
  qed
  with rtip have "the (flag rt ip) = inv" by simp
  with ⟨ip ∈ kD(rt)⟩ show "ip ∈ iD(rt)" ..
qed

lemma vD_or_iD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "ip ∈ vD(rt) ⇒ P rt ip"
  and "ip ∈ iD(rt) ⇒ P rt ip"
  shows "P rt ip"
proof -
  from ⟨ip ∈ kD(rt)⟩ have "ip ∈ vD(rt) ∪ iD(rt)"
  by (simp add: kD_is_vD_and_iD)
  thus ?thesis by (auto elim: assms(2-3))
qed

lemma proj5_eq_dhops: "∧ dip rt. dip ∈ kD(rt) ⇒ π5(the (rt dip)) = the (dhops rt dip)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj4_eq_flag: "∧ dip rt. dip ∈ kD(rt) ⇒ π4(the (rt dip)) = the (flag rt dip)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj2_eq_sqn: "∧ dip rt. dip ∈ kD(rt) ⇒ π2(the (rt dip)) = sqn rt dip"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 [simp]:
  "∧ ip rt. ip ∈ kD(rt) ⇒ sqnf rt ip = π3(the (rt ip))"
unfolding sqnf_def by auto

lemma vD_flag_val [simp]:
  "∧ dip rt. dip ∈ vD (rt) ⇒ the (flag rt dip) = val"
unfolding vD_def by clarsimp

lemma kD_update [simp]:
  "∧ rt nip v. kD (rt(nip ↦ v)) = insert nip (kD rt)"
unfolding kD_def by auto

lemma kD_empty [simp]: "kD Map.empty = {}"
unfolding kD_def by simp

```



```

lemma ip_equal_or_known [elim]:
  fixes rt ip ip'
  assumes "ip = ip'  $\vee$  ip  $\in$  kD(rt)"
  and "ip = ip'  $\implies$  P rt ip ip'"
  and "[[ ip  $\neq$  ip'; ip  $\in$  kD(rt)]]  $\implies$  P rt ip ip'"
  shows "P rt ip ip'"
  using assms by auto

```

## 2.1.4 Updating Routing Tables

Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.

### Updating Precursor Lists

```

definition addpre :: "r  $\Rightarrow$  ip set  $\Rightarrow$  r"
  where "addpre r npre  $\equiv$  let (dsn, dsk, flag, hops, nhip, pre) = r in
        (dsn, dsk, flag, hops, nhip, pre  $\cup$  npre)"

```

```

lemma proj2_addpre:
  fixes v pre
  shows " $\pi_2$ (addpre v pre) =  $\pi_2$ (v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj3_addpre:
  fixes v pre
  shows " $\pi_3$ (addpre v pre) =  $\pi_3$ (v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj4_addpre:
  fixes v pre
  shows " $\pi_4$ (addpre v pre) =  $\pi_4$ (v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj5_addpre:
  fixes v pre
  shows " $\pi_5$ (addpre v pre) =  $\pi_5$ (v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj6_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows " $\pi_6$ (addpre v npre) =  $\pi_6$ (v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj7_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows " $\pi_7$ (addpre v npre) =  $\pi_7$ (v)  $\cup$  npre"
  unfolding addpre_def by (cases v) simp

```

```

lemma addpre_empty: "addpre r {} = r"
  unfolding addpre_def by simp

```

```

lemma addpre_r:
  "addpre (dsn, dsk, fl, hops, nhip, pre) npre = (dsn, dsk, fl, hops, nhip, pre  $\cup$  npre)"
  unfolding addpre_def by simp

```

```

lemmas addpre_simps [simp] = proj2_addpre proj3_addpre proj4_addpre proj5_addpre
  proj6_addpre proj7_addpre addpre_empty addpre_r

```

```

definition addpreRT :: "rt  $\Rightarrow$  ip  $\Rightarrow$  ip set  $\rightarrow$  rt"
  where "addpreRT rt dip npre  $\equiv$ 
        map_option ( $\lambda$ s. rt (dip  $\mapsto$  addpre s npre)) ( $\sigma_{route}$ (rt, dip))"

```

```

lemma snd_addpre [simp]:
  " $\bigwedge$  dsn dsn' v pre. (dsn, snd(addpre (dsn', v) pre)) = addpre (dsn, v) pre"
  unfolding addpre_def by clarsimp

lemma proj2_addpreRT [simp]:
  fixes ip rt ip' npre
  assumes "ip  $\in$  kD rt"
  and "ip'  $\in$  kD rt"
  shows " $\pi_2$ (the (the (addpreRT rt ip' npre) ip)) =  $\pi_2$ (the (rt ip))"
  using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj3_addpreRT [simp]:
  fixes ip rt ip' npre
  assumes "ip  $\in$  kD rt"
  and "ip'  $\in$  kD rt"
  shows " $\pi_3$ (the (the (addpreRT rt ip' npre) ip)) =  $\pi_3$ (the (rt ip))"
  using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj5_addpreRT [simp]:
  " $\bigwedge$  rt dip ip npre. dip  $\in$  kD(rt)  $\implies$   $\pi_5$ (the (the (addpreRT rt dip npre) ip)) =  $\pi_5$ (the (rt ip))"
  unfolding addpreRT_def by auto

lemma flag_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip  $\in$  kD rt"
  shows "flag (the (addpreRT rt dip pre)) ip = flag rt ip"
  unfolding addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

lemma kD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip  $\in$  kD rt"
  shows "kD (the (addpreRT rt dip npre)) = kD rt"
  unfolding kD_def addpreRT_def
  using assms [THEN kD_Some]
  by clarsimp blast

lemma vD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip  $\in$  kD rt"
  shows "vD (the (addpreRT rt dip npre)) = vD rt"
  unfolding vD_def addpreRT_def
  using assms [THEN kD_Some] by clarsimp auto

lemma iD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip  $\in$  kD rt"
  shows "iD (the (addpreRT rt dip npre)) = iD rt"
  unfolding iD_def addpreRT_def
  using assms [THEN kD_Some] by clarsimp auto

lemma nhop_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip  $\in$  kD rt"
  shows "nhop (the (addpreRT rt dip pre)) ip = nhop rt ip"
  unfolding sqn_def addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

lemma sqn_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip  $\in$  kD rt"
  shows "sqn (the (addpreRT rt dip pre)) ip = sqn rt ip"
  unfolding sqn_def addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

```

```

lemma dhops_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "dhops (the (addpreRT rt dip pre)) ip = dhops rt ip"
  unfolding addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

lemma sqnf_addpreRT [simp]:
  " $\bigwedge ip\ dip. ip \in kD(rt\ \xi) \implies sqnf\ (the\ (addpreRT\ (rt\ \xi)\ ip\ npre))\ dip = sqnf\ (rt\ \xi)\ dip$ "
  unfolding sqnf_def addpreRT_def by auto

```

## Updating route entries

```

lemma in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∈ kD(rt)"
  shows "(case rt dip of None  $\implies$  en | Some r  $\implies$  es r) = es (the (rt dip))"
  using assms [THEN kD_Some] by auto

```

```

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip  $\notin$  kD(rt)"
  shows "(case rt dip of None  $\implies$  en | Some r  $\implies$  es r) = en"
  using assms [THEN kD_None] by auto

```

```

lemma rt_Some_sqn [dest]:
  fixes rt and ip dsn dsk flag hops nhip pre
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows "sqn rt ip = dsn"
  unfolding sqn_def using assms by simp

```

```

lemma not_kD_sqn [simp]:
  fixes dip rt
  assumes "dip  $\notin$  kD(rt)"
  shows "sqn rt dip = 0"
  using assms unfolding sqn_def
  by simp

```

```

definition update_arg_wf :: "r  $\implies$  bool"
where "update_arg_wf r  $\equiv$   $\pi_4(r) = val \wedge$ 
      ( $\pi_2(r) = 0$ ) = ( $\pi_3(r) = unk$ )  $\wedge$ 
      ( $\pi_3(r) = unk \implies \pi_5(r) = 1$ )"

```

```

lemma update_arg_wf_gives_cases:
  " $\bigwedge r. update\_arg\_wf\ r \implies (\pi_2(r) = 0) = (\pi_3(r) = unk)$ "
  unfolding update_arg_wf_def by simp

```

```

lemma update_arg_wf_tuples [simp]:
  " $\bigwedge nhip\ pre. update\_arg\_wf\ (0, unk, val, Suc\ 0, nhip, pre)$ "
  " $\bigwedge n\ hops\ nhip\ pre. update\_arg\_wf\ (Suc\ n, kno, val, hops, nhip, pre)$ "
  unfolding update_arg_wf_def by auto

```

```

lemma update_arg_wf_tuples' [elim]:
  " $\bigwedge n\ hops\ nhip\ pre. Suc\ 0 \leq n \implies update\_arg\_wf\ (n, kno, val, hops, nhip, pre)$ "
  unfolding update_arg_wf_def by auto

```

```

lemma wf_r_cases [intro]:
  fixes P r
  assumes "update_arg_wf r"
  and c1: " $\bigwedge nhip\ pre. P\ (0, unk, val, Suc\ 0, nhip, pre)$ "
  and c2: " $\bigwedge dsn\ hops\ nhip\ pre. dsn > 0 \implies P\ (dsn, kno, val, hops, nhip, pre)$ "
  shows "P r"
  proof -

```

```

obtain dsn dsk flag hops nhip pre
where *: "r = (dsn, dsk, flag, hops, nhip, pre)" by (cases r)
with ⟨update_arg_wf r⟩ have wf1: "flag = val"
      and wf2: "(dsn = 0) = (dsk = unk)"
      and wf3: "dsk = unk → (hops = 1)"
  unfolding update_arg_wf_def by auto
have "P (dsn, dsk, flag, hops, nhip, pre)"
proof (cases dsk)
  assume "dsk = unk"
  moreover with wf2 wf3 have "dsn = 0" and "hops = Suc 0" by auto
  ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c1)
next
  assume "dsk = kno"
  moreover with wf2 have "dsn > 0" by simp
  ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c2)
qed
with * show "P r" by simp
qed

```

definition update :: "rt ⇒ ip ⇒ r ⇒ rt"

```

where
  "update rt ip r ≡
    case σroute(rt, ip) of
      None ⇒ rt (ip ↦ r)
    | Some s ⇒
      if π2(s) < π2(r) then rt (ip ↦ addpre r (π7(s)))
      else if π2(s) = π2(r) ∧ (π5(s) > π5(r) ∨ π4(s) = inv)
      then rt (ip ↦ addpre r (π7(s)))
      else if π3(r) = unk
      then rt (ip ↦ (π2(s), snd (addpre r (π7(s))))
      else rt (ip ↦ addpre s (π7(r)))"

```

lemma update\_simps [simp]:

```

fixes r s nrt nr nr' ns rt ip
defines "s ≡ the σroute(rt, ip)"
  and "nr ≡ addpre r (π7(s))"
  and "nr' ≡ (π2(s), π3(nr), π4(nr), π5(nr), π6(nr), π7(nr))"
  and "ns ≡ addpre s (π7(r))"

```

shows

```

"[[ip ∉ kD(rt)] ⇒ update rt ip r = rt (ip ↦ r)"
"[[ip ∈ kD(rt); sqn rt ip < π2(r)] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); sqn rt ip = π2(r);
  the (dhops rt ip) > π5(r)] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); sqn rt ip = π2(r);
  flag rt ip = Some inv] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); π3(r) = unk; (π2(r) = 0) = (π3(r) = unk)] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); sqn rt ip ≥ π2(r); π3(r) = kno;
  sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val ]
  ⇒ update rt ip r = rt (ip ↦ ns)"

```

proof -

```

  assume "ip ∉ kD(rt)"
  hence "σroute(rt, ip) = None" ..
  thus "update rt ip r = rt (ip ↦ r)"
    unfolding update_def by simp
next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip < π2(r)"
  from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with ⟨sqn rt ip < π2(r)⟩ show "update rt ip r = rt (ip ↦ nr)"
    unfolding update_def nr_def s_def by auto
next
  assume "ip ∈ kD(rt)"

```

```

    and "sqn rt ip =  $\pi_2(r)$ "
    and "the (dhops rt ip) >  $\pi_5(r)$ "
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with (sqn rt ip =  $\pi_2(r)$ ) and (the (dhops rt ip) >  $\pi_5(r)$ )
    show "update rt ip r = rt (ip  $\mapsto$  nr)"
    unfolding update_def nr_def s_def by auto
next
  assume "ip  $\in$  kD(rt)"
    and "sqn rt ip =  $\pi_2(r)$ "
    and "flag rt ip = Some inv"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with (sqn rt ip =  $\pi_2(r)$ ) and (flag rt ip = Some inv)
    show "update rt ip r = rt (ip  $\mapsto$  nr)"
    unfolding update_def nr_def s_def by auto
next
  assume "ip  $\in$  kD(rt)"
    and " $\pi_3(r) = \text{unk}$ "
    and " $(\pi_2(r) = 0) = (\pi_3(r) = \text{unk})$ "
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with (( $\pi_2(r) = 0$ ) = ( $\pi_3(r) = \text{unk}$ )) and ( $\pi_3(r) = \text{unk}$ )
    show "update rt ip r = rt (ip  $\mapsto$  nr)"
    unfolding update_def nr'_def nr_def s_def
    by (cases r) simp
next
  assume "ip  $\in$  kD(rt)"
    and otherassms: "sqn rt ip  $\geq$   $\pi_2(r)$ "
    " $\pi_3(r) = \text{kno}$ "
    "sqn rt ip =  $\pi_2(r) \implies$  the (dhops rt ip)  $\leq$   $\pi_5(r) \wedge$  the (flag rt ip) = val"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with otherassms show "update rt ip r = rt (ip  $\mapsto$  ns)"
    unfolding update_def ns_def s_def by auto
qed

```

lemma update\_cases [elim]:

```

  assumes " $(\pi_2(r) = 0) = (\pi_3(r) = \text{unk})$ "
    and c1: " $\llbracket \text{ip} \notin \text{kD}(\text{rt}) \rrbracket \implies P(\text{rt}(\text{ip} \mapsto r))$ "

    and c2: " $\llbracket \text{ip} \in \text{kD}(\text{rt}); \text{sqn rt ip} < \pi_2(r) \rrbracket$ 
       $\implies P(\text{rt}(\text{ip} \mapsto \text{addpre } r(\pi_7(\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip}))))$ "
    and c3: " $\llbracket \text{ip} \in \text{kD}(\text{rt}); \text{sqn rt ip} = \pi_2(r); \text{the}(\text{dhops rt ip}) > \pi_5(r) \rrbracket$ 
       $\implies P(\text{rt}(\text{ip} \mapsto \text{addpre } r(\pi_7(\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip}))))$ "
    and c4: " $\llbracket \text{ip} \in \text{kD}(\text{rt}); \text{sqn rt ip} = \pi_2(r); \text{the}(\text{flag rt ip}) = \text{inv} \rrbracket$ 
       $\implies P(\text{rt}(\text{ip} \mapsto \text{addpre } r(\pi_7(\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip}))))$ "
    and c5: " $\llbracket \text{ip} \in \text{kD}(\text{rt}); \pi_3(r) = \text{unk} \rrbracket$ 
       $\implies P(\text{rt}(\text{ip} \mapsto (\pi_2(\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip})), \pi_3(r),$ 
         $\pi_4(r), \pi_5(r), \pi_6(r), \pi_7(\text{addpre } r(\pi_7(\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip}))))))$ "
    and c6: " $\llbracket \text{ip} \in \text{kD}(\text{rt}); \text{sqn rt ip} \geq \pi_2(r); \pi_3(r) = \text{kno};$ 
       $\text{sqn rt ip} = \pi_2(r) \implies \text{the}(\text{dhops rt ip}) \leq \pi_5(r) \wedge \text{the}(\text{flag rt ip}) = \text{val} \rrbracket$ 
       $\implies P(\text{rt}(\text{ip} \mapsto \text{addpre}(\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip}))(\pi_7(r))))$ "
  shows "(P (update rt ip r))"
  proof (cases "ip  $\in$  kD(rt)")
    assume "ip  $\notin$  kD(rt)"
    with c1 show ?thesis
      by simp
  next
    assume "ip  $\in$  kD(rt)"

```

```

moreover then obtain dsn dsk fl hops nhip pre
  where rteq: "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
moreover obtain dsn' dsk' fl' hops' nhip' pre'
  where req: "r = (dsn', dsk', fl', hops', nhip', pre')"
    by (cases r) metis
ultimately show ?thesis
  using ⟨(π2(r) = 0) = (π3(r) = unk)⟩
    c2 [OF ⟨ip ∈ kD(rt)⟩]
    c3 [OF ⟨ip ∈ kD(rt)⟩]
    c4 [OF ⟨ip ∈ kD(rt)⟩]
    c5 [OF ⟨ip ∈ kD(rt)⟩]
    c6 [OF ⟨ip ∈ kD(rt)⟩]
  unfolding update_def sqn_def by auto
qed

```

lemma update\_cases\_kD:

```

assumes "(π2(r) = 0) = (π3(r) = unk)"
  and "ip ∈ kD(rt)"
  and c2: "sqn rt ip < π2(r) ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
  and c3: "⟦sqn rt ip = π2(r); the (dhops rt ip) > π5(r)⟧
    ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
  and c4: "⟦sqn rt ip = π2(r); the (flag rt ip) = inv⟧
    ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
  and c5: "π3(r) = unk ⇒ P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r),
    π4(r), π5(r), π6(r),
    π7(addpre r (π7(the σroute(rt, ip))))))"
  and c6: "⟦sqn rt ip ≥ π2(r); π3(r) = kno;
    sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val⟧
    ⇒ P (rt (ip ↦ addpre (the σroute(rt, ip)) (π7(r))))"
shows "(P (update rt ip r))"
using assms(1) proof (rule update_cases)
  assume "sqn rt ip < π2(r)"
  thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))" by (rule c2)
next
  assume "sqn rt ip = π2(r)"
  and "the (dhops rt ip) > π5(r)"
  thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))"
  by (rule c3)
next
  assume "sqn rt ip = π2(r)"
  and "the (flag rt ip) = inv"
  thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))"
  by (rule c4)
next
  assume "π3(r) = unk"
  thus "P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r), π4(r), π5(r), π6(r),
    π7(addpre r (π7(the (rt ip))))))"
  by (rule c5)
next
  assume "sqn rt ip ≥ π2(r)"
  and "π3(r) = kno"
  and "sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val"
  thus "P (rt (ip ↦ addpre (the (rt ip)) (π7(r))))"
  by (rule c6)
qed (simp add: ⟨ip ∈ kD(rt)⟩)

```

lemma in\_kD\_after\_update [simp]:

```

fixes rt nip dsn dsk flag hops nhip pre
shows "kD (update rt nip (dsn, dsk, flag, hops, nhip, pre)) = insert nip (kD rt)"
unfolding update_def
by (cases "rt nip") auto

```

lemma nhop\_of\_update [simp]:

```

fixes rt dip dsn dsk flag hops nhip
assumes "rt  $\neq$  update rt dip (dsn, dsk, flag, hops, nhip, {})"
shows "the (nhop (update rt dip (dsn, dsk, flag, hops, nhip, {})) dip) = nhip"
proof -
from assms
have update_neq: " $\bigwedge v. rt\ dip = Some\ v \implies$ 
  update rt dip (dsn, dsk, flag, hops, nhip, {})
   $\neq$  rt(dip  $\mapsto$  addpre (the (rt dip)) ( $\pi_7$  (dsn, dsk, flag, hops, nhip, {})))"
  by auto
show ?thesis
proof (cases "rt dip = None")
  assume "rt dip = None"
  thus "?thesis" unfolding update_def by clarsimp
next
  assume "rt dip  $\neq$  None"
  then obtain v where "rt dip = Some v" by (metis not_None_eq)
  with update_neq [OF this] show ?thesis
  unfolding update_def by auto
qed
qed

lemma sqn_if_updated:
fixes rip v rt ip
shows "sqn ( $\lambda x. if\ x = rip\ then\ Some\ v\ else\ rt\ x$ ) ip
  = (if ip = rip then  $\pi_2(v)$  else sqn rt ip)"
unfolding sqn_def by simp

lemma update_sqn [simp]:
fixes rt dip rip dsn dsk hops nhip pre
assumes "(dsn = 0) = (dsk = unk)"
shows "sqn rt dip  $\leq$  sqn (update rt rip (dsn, dsk, val, hops, nhip, pre)) dip"
proof (rule update_cases)
  show "( $\pi_2$  (dsn, dsk, val, hops, nhip, pre) = 0) = ( $\pi_3$  (dsn, dsk, val, hops, nhip, pre) = unk)"
  by simp (rule assms)
qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
fixes rt ip ip' dsn dsk flag hops nhip pre
assumes "1  $\leq$  hops"
shows "sqn rt ip  $\leq$  sqn (update rt ip' (dsn, dsk, flag, hops, nhip, pre)) ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
fixes rt dsn dsk flag hops ip rip nhip pre
assumes ex: " $\forall ip \in kD\ rt. the\ (dhops\ rt\ ip) \geq 1$ "
  and ip: "(ip = rip  $\wedge$  Suc 0  $\leq$  hops)  $\vee$  (ip  $\neq$  rip  $\wedge$  ip  $\in kD\ rt)"
shows "Suc 0  $\leq$  the (dhops (update rt rip (dsn, dsk, flag, hops, nhip, pre)) ip)"
using ip proof
  assume "ip = rip  $\wedge$  Suc 0  $\leq$  hops" thus ?thesis
  unfolding update_def using ex
  by (cases "rip  $\in kD\ rt$ ") (drule(1) bspec, auto)
next
  assume "ip  $\neq$  rip  $\wedge$  ip  $\in kD\ rt$ " thus ?thesis
  using ex unfolding update_def
  by (cases "rip  $\in kD\ rt$ ") auto
qed

lemma update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip pre
assumes "ip  $\neq$  dip"
shows "(update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = rt ip"
using assms unfolding update_def
by (clarsimp split: option.split)$ 
```

```

lemma nhop_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = nhop rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma dhops_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = dhops rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
  "∧rt ip dsn dsk flag hops nhip pre. sqn (rt(ip ↦ v)) ip = π2(v)"
  unfolding sqn_def by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip
  assumes "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nhip, {})) dip) = hops"
  using assms unfolding update_def
  by (clarsimp split: option.split_asm option.split split_if_asm) auto

lemma nhop_update_unk_val [simp]:
  "∧rt dip ip dsn hops npre.
  the (nhop (update rt dip (dsn, unk, val, hops, ip, npre)) dip) = ip"
  unfolding update_def by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
  assumes "update rt dip (dsn, dsk, flg, hops, sip, {}) ≠ rt"
  shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
  using assms unfolding update_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma update_rt_split_asm:
  "∧rt ip dsn dsk flag hops sip.
  P (update rt ip (dsn, dsk, flag, hops, sip, {}))
  =
  (¬(rt = update rt ip (dsn, dsk, flag, hops, sip, {})) ∧ ¬P rt
  ∨ rt ≠ update rt ip (dsn, dsk, flag, hops, sip, {})
  ∧ ¬P (update rt ip (dsn, dsk, flag, hops, sip, {})))"
  by auto

lemma sqn_update [simp]: "∧rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
  ⇒ sqn (update rt dip (dsn, kno, flg, hops, sip, {})) dip = dsn"
  unfolding update_def by (clarsimp split: option.split split_if_asm) auto

lemma sqnf_update [simp]: "∧rt dip dsn dsk flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
  ⇒ sqnf (update rt dip (dsn, dsk, flg, hops, sip, {})) dip = dsk"
  unfolding update_def sqnf_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma update_kno_dsn_greater_zero:
  "∧rt dip ip dsn hops npre. 1 ≤ dsn ⇒ 1 ≤ (sqn (update rt dip (dsn, kno, val, hops, ip, npre)) dip)"
  unfolding update_def
  by (clarsimp split: option.splits)

lemma proj3_update [simp]: "∧rt dip dsn dsk flg hops sip.

```



```

rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
⇒ π3(the (update rt dip (dsn, dsk, flg, hops, sip, {}) dip)) = dsk"
unfolding update_def sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

lemma nhop_update_changed_kno_val [simp]: "∧rt ip dsn dsk hops nhip.
rt ≠ update rt ip (dsn, kno, val, hops, nhip, {})
⇒ the (nhop (update rt ip (dsn, kno, val, hops, nhip, {})) ip) = nhip"
using assms unfolding update_def
by (clarsimp split: option.split_asm option.split split_if_asm) auto

lemma flag_update [simp]: "∧rt dip dsn flg hops sip.
rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
⇒ the (flag (update rt dip (dsn, kno, flg, hops, sip, {})) dip) = flg"
unfolding update_def
by (clarsimp split: option.split split_if_asm) auto

lemma the_flag_Some [dest!]:
  fixes ip rt
  assumes "the (flag rt ip) = x"
  and "ip ∈ kD rt"
  shows "flag rt ip = Some x"
  using assms by auto

lemma kD_update_unchanged [dest]:
  fixes rt dip dsn dsk flag hops nhip pre
  assumes "rt = update rt dip (dsn, dsk, flag, hops, nhip, pre)"
  shows "dip ∈ kD(rt)"
  proof -
    have "dip ∈ kD(update rt dip (dsn, dsk, flag, hops, nhip, pre))" by simp
    with assms show ?thesis by simp
  qed

lemma nhop_update [simp]: "∧rt dip dsn dsk flg hops sip.
rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
⇒ the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
unfolding update_def sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

lemma sqn_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqn rt ip"
  using assms unfolding update_def sqn_def
  by (clarsimp split: option.splits) auto

lemma sqnf_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "sqnf (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqnf rt ip"
  using assms unfolding update_def sqnf_def
  by (clarsimp split: option.splits) auto

lemma vD_update_val [dest]:
  "∧dip rt dip' dsn dsk hops nhip pre.
dip ∈ vD(update rt dip' (dsn, dsk, val, hops, nhip, pre)) ⇒ (dip ∈ vD(rt) ∨ dip=dip'"
  unfolding update_def vD_def by (clarsimp split: option.split_asm split_if_asm)

```

## Invalidating route entries

```

definition invalidate :: "rt ⇒ (ip ⇒ sqn) ⇒ rt"
where "invalidate rt dests ≡
λip. case (rt ip, dests ip) of
  (None, _) ⇒ None

```

```

| (Some s, None) ⇒ Some s
| (Some (_, dsk, _, hops, nhip, pre), Some rsn) ⇒
  Some (rsn, dsk, inv, hops, nhip, pre)"

lemma proj3_invalidate [simp]:
  "∧dip. π3(the ((invalidate rt dests) dip)) = π3(the (rt dip))"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma proj5_invalidate [simp]:
  "∧dip. π5(the ((invalidate rt dests) dip)) = π5(the (rt dip))"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma proj6_invalidate [simp]:
  "∧dip. π6(the ((invalidate rt dests) dip)) = π6(the (rt dip))"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma proj7_invalidate [simp]:
  "∧dip. π7(the ((invalidate rt dests) dip)) = π7(the (rt dip))"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_kD_inv [simp]:
  "∧rt dests. kD (invalidate rt dests) = kD rt"
  unfolding invalidate_def kD_def
  by (simp split: option.split)

lemma invalidate_sqn:
  fixes rt dip dests
  assumes "∀rsn. dests dip = Some rsn → sqn rt dip ≤ rsn"
  shows "sqn rt dip ≤ sqn (invalidate rt dests) dip"
  proof (cases "dip ∈ kD(rt)")
    assume "¬ dip ∈ kD(rt)"
    hence "dip ∈ kD(rt)" by simp
    then obtain dsn dsk flag hops nhip pre where "rt dip = Some (dsn, dsk, flag, hops, nhip, pre)"
      by (metis kD_Some)
    with assms show "sqn rt dip ≤ sqn (invalidate rt dests) dip"
      by (cases "dests dip") (auto simp add: invalidate_def sqn_def)
  qed simp

lemma sqn_invalidate_in_dests [simp]:
  fixes dests ipa rsn rt
  assumes "dests ipa = Some rsn"
    and "ipa ∈ kD(rt)"
  shows "sqn (invalidate rt dests) ipa = rsn"
  unfolding invalidate_def sqn_def
  using assms(1) assms(2) [THEN kD_Some]
  by clarsimp

lemma dhops_invalidate [simp]:
  "∧dip. the (dhops (invalidate rt dests) dip) = the (dhops rt dip)"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma sqnf_invalidate [simp]:
  "∧dip. sqnf (invalidate (rt ξ) (dests ξ)) dip = sqnf (rt ξ) dip"
  unfolding sqnf_def invalidate_def by (clarsimp split: option.split)

lemma nhop_invalidate [simp]:
  "∧dip. the (nhop (invalidate (rt ξ) (dests ξ)) dip) = the (nhop (rt ξ) dip)"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
  fixes rt dests dip
  assumes "dip ∉ dom(dests)"
  shows "invalidate rt dests dip = rt dip"
  using assms unfolding invalidate_def

```

```

by (clarsimp split: option.split_asm)

lemma invalidate_none [simp]:
  fixes rt dests dip
  assumes "dip ∉ kD(rt)"
  shows "invalidate rt dests dip = None"
  using assms unfolding invalidate_def by clarsimp

lemma vD_invalidate_vD_not_dests:
  "∧ dip rt dests. dip ∈ vD(invalidate rt dests) ⇒ dip ∈ vD(rt) ∧ dests dip = None"
  unfolding invalidate_def vD_def
  by (clarsimp split: option.split_asm)

lemma sqn_invalidate_not_in_dests [simp]:
  fixes dests dip rt
  assumes "dip ∉ dom(dests)"
  shows "sqn (invalidate rt dests) dip = sqn rt dip"
  using assms unfolding sqn_def by simp

lemma invalidate_changes:
  fixes rt dests dip dsn dsk flag hops nhip pre
  assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows " dsn = (case dests dip of None ⇒ π2(the (rt dip)) | Some rsn ⇒ rsn)
    ∧ dsk = π3(the (rt dip))
    ∧ flag = (if dests dip = None then π4(the (rt dip)) else inv)
    ∧ hops = π5(the (rt dip))
    ∧ nhip = π6(the (rt dip))
    ∧ pre = π7(the (rt dip))"
  using assms unfolding invalidate_def
  by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: "∧ dip rt dests. dip ∈ kD (rt)
  ⇒ π3(the (invalidate rt dests dip)) = π3(the (rt dip))"
  by (clarsimp simp: invalidate_def kD_def split: option.split)

lemma dests_iD_invalidate [simp]:
  assumes "dests ip = Some rsn"
  and "ip ∈ kD(rt)"
  shows "ip ∈ iD(invalidate rt dests)"
  using assms(1) assms(2) [THEN kD_Some] unfolding invalidate_def iD_def
  by (clarsimp split: option.split)

```

### 2.1.5 Route Requests

Generate a fresh route request identifier.

```

definition nrreqid :: "(ip × rreqid) set ⇒ ip ⇒ rreqid"
  where "nrreqid rreqs ip ≡ Max ({n. (ip, n) ∈ rreqs} ∪ {0}) + 1"

```

### 2.1.6 Queued Packets

Functions for sending data packets.

```

type_synonym store = "ip → (p × data list)"

```

```

definition sigma_queue :: "store ⇒ ip ⇒ data list" ("σqueue'(_, _)'")
  where "σqueue(store, dip) ≡ case store dip of None ⇒ [] | Some (p, q) ⇒ q"

```

```

definition qD :: "store ⇒ ip set"
  where "qD ≡ dom"

```

```

definition add :: "data ⇒ ip ⇒ store ⇒ store"
  where "add d dip store ≡ case store dip of
    None ⇒ store (dip ↦ (req, [d]))

```

|  $\text{Some } (p, q) \Rightarrow \text{store } (\text{dip} \mapsto (p, q @ [d]))$ "

```
lemma qD_add [simp]:
  fixes d dip store
  shows "qD(add d dip store) = insert dip (qD store)"
  unfolding add_def Let_def qD_def
  by (clarsimp split: option.split)
```

```
definition drop :: "ip  $\Rightarrow$  store  $\rightarrow$  store"
  where "drop dip store  $\equiv$ 
    map_option ( $\lambda(p, q). \text{if } \text{tl } q = [] \text{ then store } (\text{dip} := \text{None})$ 
               else store ( $\text{dip} \mapsto (p, \text{tl } q)$ )) (store dip)"
```

```
definition sigma_p_flag :: "store  $\Rightarrow$  ip  $\rightarrow$  p" (" $\sigma_{p\text{-flag}}'(\_, \_)$ ")
  where " $\sigma_{p\text{-flag}}(\text{store}, \text{dip}) \equiv \text{map\_option fst } (\text{store dip})$ "
```

```
definition unsetRRF :: "store  $\Rightarrow$  ip  $\Rightarrow$  store"
  where "unsetRRF store dip  $\equiv$  case store dip of
    None  $\Rightarrow$  store
    | Some (p, q)  $\Rightarrow$  store ( $\text{dip} \mapsto (\text{noreq}, q)$ )"
```

```
definition setRRF :: "store  $\Rightarrow$  (ip  $\rightarrow$  sqn)  $\Rightarrow$  store"
  where "setRRF store dests  $\equiv$   $\lambda$ dip. if dests dip = None then store dip
    else map_option ( $\lambda(\_, q). (\text{req}, q)$ ) (store dip)"
```

### 2.1.7 Comparison with the original technical report

The major differences with the AODV technical report of Fehnker et al are:

1. *nhop* is partial, thus a ‘*the*’ is needed, similarly for *dhops* and *addpreRT*.
2. *precs* is partial.
3.  $\sigma_{p\text{-flag}}(\text{store}, \text{dip})$  is partial.
4. The routing table (*rt*) is modelled as a map ( $\text{ip} \Rightarrow r \text{ option}$ ) rather than a set of 7-tuples, likewise, the *r* is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address (*dip*) is taken from the argument to the function, rather than a part of the result. Well-definedness then follows from the structure of the type and more related facts are available automatically, rather than having to be acquired through tedious proofs.
5. Similar remarks hold for the *dests* mapping passed to *invalidate*, and *store*.

end

## 2.2 AODV protocol messages

```
theory B_Aodv_Message
  imports B_Fwdrreps
  begin
```

```
datatype msg =
  Rreq nat rreqid ip sqn k ip sqn ip
  | Rrep nat ip sqn ip ip
  | Rerr "ip  $\rightarrow$  sqn" ip
  | Newpkt data ip
  | Pkt data ip ip
```

```
instantiation msg :: msg
begin
```

```
definition newpkt_def [simp]: "newpkt  $\equiv$   $\lambda(d, \text{dip}). \text{Newpkt } d \text{ dip}"$ 
definition eq_newpkt_def: "eq_newpkt m  $\equiv$  case m of Newpkt d dip  $\Rightarrow$  True | _  $\Rightarrow$  False"
```

```
instance by intro_classes (simp add: eq_newpkt_def)
```

end

The *msg* type models the different messages used within AODV. The instantiation as a *msg* is a technicality due to the special treatment of *newpkt* messages in the AWN SOS rules. This use of classes allows a clean separation of the AWN-specific definitions and these AODV-specific definitions.

```
definition rreq :: "nat × rreqid × ip × sqn × k × ip × sqn × ip ⇒ msg"
  where "rreq ≡ λ(hops, rreqid, dip, dsn, dsk, oip, osn, sip).
        Rreq hops rreqid dip dsn dsk oip osn sip"
```

```
lemma rreq_simp [simp]:
  "rreq(hops, rreqid, dip, dsn, dsk, oip, osn, sip) = Rreq hops rreqid dip dsn dsk oip osn sip"
  unfolding rreq_def by simp
```

```
definition rrep :: "nat × ip × sqn × ip × ip ⇒ msg"
  where "rrep ≡ λ(hops, dip, dsn, oip, sip). Rrep hops dip dsn oip sip"
```

```
lemma rrep_simp [simp]:
  "rrep(hops, dip, dsn, oip, sip) = Rrep hops dip dsn oip sip"
  unfolding rrep_def by simp
```

```
definition rerr :: "(ip → sqn) × ip ⇒ msg"
  where "rerr ≡ λ(dests, sip). Rerr dests sip"
```

```
lemma rerr_simp [simp]:
  "rerr(dests, sip) = Rerr dests sip"
  unfolding rerr_def by simp
```

```
lemma not_eq_newpkt_rreq [simp]: "¬eq_newpkt (Rreq hops rreqid dip dsn dsk oip osn sip)"
  unfolding eq_newpkt_def by simp
```

```
lemma not_eq_newpkt_rrep [simp]: "¬eq_newpkt (Rrep hops dip dsn oip sip)"
  unfolding eq_newpkt_def by simp
```

```
lemma not_eq_newpkt_rerr [simp]: "¬eq_newpkt (Rerr dests sip)"
  unfolding eq_newpkt_def by simp
```

```
lemma not_eq_newpkt_pkt [simp]: "¬eq_newpkt (Pkt d dip sip)"
  unfolding eq_newpkt_def by simp
```

```
definition pkt :: "data × ip × ip ⇒ msg"
  where "pkt ≡ λ(d, dip, sip). Pkt d dip sip"
```

```
lemma pkt_simp [simp]:
  "pkt(d, dip, sip) = Pkt d dip sip"
  unfolding pkt_def by simp
```

end

## 2.3 The AODV protocol

```
theory B_Aodv
  imports B_Aodv_Data B_Aodv_Message
          "../.../AWN/AWN_SOS_Labels" "../.../AWN/AWN_Invariants"
begin
```

### 2.3.1 Data state

```
record state =
  ip      :: "ip"
  sn      :: "sqn"
  rt      :: "rt"
  rreqs   :: "(ip × rreqid) set"
  store   :: "store"
```

```

msg    :: "msg"
data   :: "data"
destds :: "ip  $\rightarrow$  sqn"
pre    :: "ip set"
rreqid :: "rreqid"
dip    :: "ip"
oip    :: "ip"
hops   :: "nat"
dsn    :: "sqn"
dsk    :: "k"
osn    :: "sqn"
sip    :: "ip"

```

abbreviation aadv\_init :: "ip  $\Rightarrow$  state"

```

where "aadv_init i  $\equiv$  ( $\{$ 
  ip = i,
  sn = 1,
  rt = empty,
  rreqs = {},
  store = empty,

  msg    = (SOME x. True),
  data   = (SOME x. True),
  destds = (SOME x. True),
  pre    = (SOME x. True),
  rreqid = (SOME x. True),
  dip    = (SOME x. True),
  oip    = (SOME x. True),
  hops   = (SOME x. True),
  dsn    = (SOME x. True),
  dsk    = (SOME x. True),
  osn    = (SOME x. True),
  sip    = (SOME x. x  $\neq$  i)
 $\}$ "

```

lemma some\_neq\_not\_eq [simp]: " $\neg((\text{SOME } x :: \text{nat}. x \neq i) = i)$ "  
 by (subst some\_eq\_ex) (metis zero\_neq\_numeral)

definition clear\_locals :: "state  $\Rightarrow$  state"

```

where "clear_locals  $\xi$  =  $\xi$  ( $\{$ 
  msg    := (SOME x. True),
  data   := (SOME x. True),
  destds := (SOME x. True),
  pre    := (SOME x. True),
  rreqid := (SOME x. True),
  dip    := (SOME x. True),
  oip    := (SOME x. True),
  hops   := (SOME x. True),
  dsn    := (SOME x. True),
  dsk    := (SOME x. True),
  osn    := (SOME x. True),
  sip    := (SOME x. x  $\neq$  ip  $\xi$ )
 $\}$ "

```

lemma clear\_locals\_sip\_not\_ip [simp]: " $\neg(\text{sip } (\text{clear\_locals } \xi) = \text{ip } \xi)$ "  
 unfolding clear\_locals\_def by simp

lemma clear\_locals\_but\_not\_globals [simp]:

```

"ip (clear_locals  $\xi$ ) = ip  $\xi$ "
"sn (clear_locals  $\xi$ ) = sn  $\xi$ "
"rt (clear_locals  $\xi$ ) = rt  $\xi$ "
"rreqs (clear_locals  $\xi$ ) = rreqs  $\xi$ "
"store (clear_locals  $\xi$ ) = store  $\xi$ "

```

unfolding `clear_locals_def` by `auto`

### 2.3.2 Auxilliary message handling definitions

**definition** `is_newpkt`

where "`is_newpkt`  $\xi$   $\equiv$  case `msg`  $\xi$  of  
    `Newpkt data' dip'`  $\Rightarrow$  {  $\xi$ (`data := data'`, `dip := dip'`) }  
    | `_`  $\Rightarrow$  {}"

**definition** `is_pkt`

where "`is_pkt`  $\xi$   $\equiv$  case `msg`  $\xi$  of  
    `Pkt data' dip' oip'`  $\Rightarrow$  {  $\xi$ (`data := data'`, `dip := dip'`, `oip := oip'`) }  
    | `_`  $\Rightarrow$  {}"

**definition** `is_rreq`

where "`is_rreq`  $\xi$   $\equiv$  case `msg`  $\xi$  of  
    `Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip'`  $\Rightarrow$   
    {  $\xi$ (`hops := hops'`, `rreqid := rreqid'`, `dip := dip'`, `dsn := dsn'`,  
        `dsk := dsk'`, `oip := oip'`, `osn := osn'`, `sip := sip'`) }  
    | `_`  $\Rightarrow$  {}"

**lemma** `is_rreq_asm [dest!]`:

assumes " $\xi' \in$  `is_rreq`  $\xi$ "

shows " $(\exists$  `hops' rreqid' dip' dsn' dsk' oip' osn' sip'`.  
    `msg`  $\xi =$  `Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip'`  $\wedge$   
     $\xi' = \xi$ (`hops := hops'`, `rreqid := rreqid'`, `dip := dip'`, `dsn := dsn'`,  
        `dsk := dsk'`, `oip := oip'`, `osn := osn'`, `sip := sip'`)")"

using `assms unfolding is_rreq_def`

by (cases "`msg`  $\xi$ ") `simp_all`

**definition** `is_rrep`

where "`is_rrep`  $\xi$   $\equiv$  case `msg`  $\xi$  of  
    `Rrep hops' dip' dsn' oip' sip'`  $\Rightarrow$   
    {  $\xi$ (`hops := hops'`, `dip := dip'`, `dsn := dsn'`, `oip := oip'`, `sip := sip'`) }  
    | `_`  $\Rightarrow$  {}"

**lemma** `is_rrep_asm [dest!]`:

assumes " $\xi' \in$  `is_rrep`  $\xi$ "

shows " $(\exists$  `hops' dip' dsn' oip' sip'`.  
    `msg`  $\xi =$  `Rrep hops' dip' dsn' oip' sip'`  $\wedge$   
     $\xi' = \xi$ (`hops := hops'`, `dip := dip'`, `dsn := dsn'`, `oip := oip'`, `sip := sip'`)")"

using `assms unfolding is_rrep_def`

by (cases "`msg`  $\xi$ ") `simp_all`

**definition** `is_rerr`

where "`is_rerr`  $\xi$   $\equiv$  case `msg`  $\xi$  of  
    `Rerr dests' sip'`  $\Rightarrow$  {  $\xi$ (`dests := dests'`, `sip := sip'`) }  
    | `_`  $\Rightarrow$  {}"

**lemma** `is_rerr_asm [dest!]`:

assumes " $\xi' \in$  `is_rerr`  $\xi$ "

shows " $(\exists$  `dests' sip'`.  
    `msg`  $\xi =$  `Rerr dests' sip'`  $\wedge$   
     $\xi' = \xi$ (`dests := dests'`, `sip := sip'`)")"

using `assms unfolding is_rerr_def`

by (cases "`msg`  $\xi$ ") `simp_all`

**lemmas** `is_msg_defs =`

`is_rerr_def is_rrep_def is_rreq_def is_pkt_def is_newpkt_def`

**lemma** `is_msg_inv_ip [simp]`:

" $\xi' \in$  `is_rerr`  $\xi$   $\implies$  `ip`  $\xi' =$  `ip`  $\xi$ "

" $\xi' \in$  `is_rrep`  $\xi$   $\implies$  `ip`  $\xi' =$  `ip`  $\xi$ "

" $\xi' \in$  `is_rreq`  $\xi$   $\implies$  `ip`  $\xi' =$  `ip`  $\xi$ "

```

"ξ' ∈ is_pkt ξ    ⇒ ip ξ' = ip ξ"
"ξ' ∈ is_newpkt ξ ⇒ ip ξ' = ip ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

```

```

lemma is_msg_inv_sn [simp]:
"ξ' ∈ is_rerr ξ    ⇒ sn ξ' = sn ξ"
"ξ' ∈ is_rrep ξ    ⇒ sn ξ' = sn ξ"
"ξ' ∈ is_rreq ξ    ⇒ sn ξ' = sn ξ"
"ξ' ∈ is_pkt ξ     ⇒ sn ξ' = sn ξ"
"ξ' ∈ is_newpkt ξ ⇒ sn ξ' = sn ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

```

```

lemma is_msg_inv_rt [simp]:
"ξ' ∈ is_rerr ξ    ⇒ rt ξ' = rt ξ"
"ξ' ∈ is_rrep ξ    ⇒ rt ξ' = rt ξ"
"ξ' ∈ is_rreq ξ    ⇒ rt ξ' = rt ξ"
"ξ' ∈ is_pkt ξ     ⇒ rt ξ' = rt ξ"
"ξ' ∈ is_newpkt ξ ⇒ rt ξ' = rt ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

```

```

lemma is_msg_inv_rreqs [simp]:
"ξ' ∈ is_rerr ξ    ⇒ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_rrep ξ    ⇒ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_rreq ξ    ⇒ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_pkt ξ     ⇒ rreqs ξ' = rreqs ξ"
"ξ' ∈ is_newpkt ξ ⇒ rreqs ξ' = rreqs ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

```

```

lemma is_msg_inv_store [simp]:
"ξ' ∈ is_rerr ξ    ⇒ store ξ' = store ξ"
"ξ' ∈ is_rrep ξ    ⇒ store ξ' = store ξ"
"ξ' ∈ is_rreq ξ    ⇒ store ξ' = store ξ"
"ξ' ∈ is_pkt ξ     ⇒ store ξ' = store ξ"
"ξ' ∈ is_newpkt ξ ⇒ store ξ' = store ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

```

```

lemma is_msg_inv_sip [simp]:
"ξ' ∈ is_pkt ξ     ⇒ sip ξ' = sip ξ"
"ξ' ∈ is_newpkt ξ ⇒ sip ξ' = sip ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp+)+

```

### 2.3.3 The protocol process

```

datatype pseqp =
  PAadv
  | PNewPkt
  | PPkt
  | PRreq
  | PRrep
  | PRerr

```

```

fun nat_of_seqp :: "pseqp ⇒ nat"
where
  "nat_of_seqp PAadv = 1"
| "nat_of_seqp PPkt = 2"
| "nat_of_seqp PNewPkt = 3"
| "nat_of_seqp PRreq = 4"
| "nat_of_seqp PRrep = 5"

```



```
| "nat_of_seqp PRerr = 6"
```

```
instantiation "pseqp" :: ord
```

```
begin
```

```
definition less_eq_seqp [iff]: "l1 ≤ l2 = (nat_of_seqp l1 ≤ nat_of_seqp l2)"
```

```
definition less_seqp [iff]: "l1 < l2 = (nat_of_seqp l1 < nat_of_seqp l2)"
```

```
instance ..
```

```
end
```

```
abbreviation AODV
```

```
where
```

```
"AODV ≡ λ_. [[clear_locals]] call(PAodv)"
```

```
abbreviation PKT
```

```
where
```

```
"PKT args ≡
```

```
[[ξ. let (data, dip, oip) = args ξ in  
  (clear_locals ξ) (| data := data, dip := dip, oip := oip |)]  
  call(PPkt)"
```

```
abbreviation NEWPKT
```

```
where
```

```
"NEWPKT args ≡  
[[ξ. let (data, dip) = args ξ in  
  (clear_locals ξ) (| data := data, dip := dip |)]  
  call(PNewPkt)"
```

```
abbreviation RREQ
```

```
where
```

```
"RREQ args ≡  
[[ξ. let (hops, rreqid, dip, dsn, dsk, oip, osn, sip) = args ξ in  
  (clear_locals ξ) (| hops := hops, rreqid := rreqid, dip := dip,  
    dsn := dsn, dsk := dsk, oip := oip,  
    osn := osn, sip := sip |)]  
  call(PRreq)"
```

```
abbreviation RREP
```

```
where
```

```
"RREP args ≡  
[[ξ. let (hops, dip, dsn, oip, sip) = args ξ in  
  (clear_locals ξ) (| hops := hops, dip := dip, dsn := dsn,  
    oip := oip, sip := sip |)]  
  call(PRrep)"
```

```
abbreviation RERR
```

```
where
```

```
"RERR args ≡  
[[ξ. let (dests, sip) = args ξ in  
  (clear_locals ξ) (| dests := dests, sip := sip |)]  
  call(PRerr)"
```

```
fun ΓAODV :: "(state, msg, pseqp, pseqp label) seqp_env"
```

```
where
```

```
"ΓAODV PAodv = labelled PAodv (  
  receive(λmsg' ξ. ξ (| msg := msg' |)).  
  ( |is_newpkt> NEWPKT(λξ. (data ξ, ip ξ))  
    ⊕ |is_pkt> PKT(λξ. (data ξ, dip ξ, oip ξ))  
    ⊕ |is_rreq>  
      [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, {})|)]  
      RREQ(λξ. (hops ξ, rreqid ξ, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, sip ξ))  
    ⊕ |is_rrep>  
      [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, {})|)]  
      RREP(λξ. (hops ξ, dip ξ, dsn ξ, oip ξ, sip ξ))  
    ⊕ |is_rerr>
```

```

    [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, { }) )]]
    RERR(λξ. (dests ξ, sip ξ))
  )
⊕ ⟨λξ. { ξ (| dip := dip ) | dip. dip ∈ qD(store ξ) ∩ vD(rt ξ) }⟩
  [[ξ. ξ (| data := hd(σqueue(store ξ, dip ξ)) )]]
  unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, ip ξ)).
  [[ξ. ξ (| store := the (drop (dip ξ) (store ξ)) )]]
  AODV()
  ▷ [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
    then Some (inc (sqn (rt ξ) rip)) else None) )]]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
    [[ξ. ξ (| pre := ⋃{ the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
    [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
    then (dests ξ) rip else None) )]]
    groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)). AODV()
⊕ ⟨λξ. { ξ (| dip := dip )
  | dip. dip ∈ qD(store ξ) - vD(rt ξ) ∧ the (σp-flag(store ξ, dip)) = req }⟩
  [[ξ. ξ (| store := unsetRRF (store ξ) (dip ξ) )]]
  [[ξ. ξ (| sn := inc (sn ξ) )]]
  [[ξ. ξ (| rreqid := nrreqid (rreqs ξ) (ip ξ) )]]
  [[ξ. ξ (| rreqs := rreqs ξ ∪ {(ip ξ, rreqid ξ)} )]]
  broadcast(λξ. rreq(0, rreqid ξ, dip ξ, sqn (rt ξ) (dip ξ), sqnf (rt ξ) (dip ξ),
    ip ξ, sn ξ, ip ξ)). AODV()"

/ "ΓAODV PNewPkt = labelled PNewPkt (
  ⟨ξ. dip ξ = ip ξ⟩
  deliver(λξ. data ξ).AODV()
⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
  [[ξ. ξ (| store := add (data ξ) (dip ξ) (store ξ) )]]
  AODV()"

/ "ΓAODV PPkt = labelled PPkt (
  ⟨ξ. dip ξ = ip ξ⟩
  deliver(λξ. data ξ).AODV()
⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
  (
    ⟨ξ. dip ξ ∈ vD (rt ξ)⟩
    unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, oip ξ)).AODV()
    ▷
    [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
    then Some (inc (sqn (rt ξ) rip)) else None) )]]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
    [[ξ. ξ (| pre := ⋃{ the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
    [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
    then (dests ξ) rip else None) )]]
    groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
⊕ ⟨ξ. dip ξ ∉ vD (rt ξ)⟩
  (
    ⟨ξ. dip ξ ∈ iD (rt ξ)⟩
    groupcast(λξ. the (precs (rt ξ) (dip ξ)),
      λξ. rerr([dip ξ ↦ sqn (rt ξ) (dip ξ)], ip ξ)). AODV()
⊕ ⟨ξ. dip ξ ∉ iD (rt ξ)⟩
  AODV()
  )
  )
  )"

/ "ΓAODV PRreq = labelled PRreq (
  ⟨ξ. (oip ξ, rreqid ξ) ∈ rreqs ξ⟩
  AODV()
⊕ ⟨ξ. (oip ξ, rreqid ξ) ∉ rreqs ξ⟩
  [[ξ. ξ (| rt := update (rt ξ) (oip ξ) (osn ξ, kno, val, hops ξ + 1, sip ξ, { }) )]]
  [[ξ. ξ (| rreqs := rreqs ξ ∪ {(oip ξ, rreqid ξ)} )]]

```

```

(
  ⟨ξ. dip ξ = ip ξ⟩
  [[ξ. ξ (| sn := max (sn ξ) (dsn ξ) )]]
  unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(0, dip ξ, sn ξ, oip ξ, ip ξ)).AODV()
  ▷
  [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
    then Some (inc (sqn (rt ξ) rip)) else None) )]]
  [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
  [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
  [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
  [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
    then (dests ξ) rip else None) )]]
  groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
  ⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
  (
    ⟨ξ. dip ξ ∈ vD (rt ξ) ∧ dsn ξ ≤ sqn (rt ξ) (dip ξ) ∧ sqnf (rt ξ) (dip ξ) = kno⟩
    [[ξ. ξ (| rt := the (addpreRT (rt ξ) (dip ξ) {sip ξ}) )]]
    [[ξ. ξ (| rt := the (addpreRT (rt ξ) (oip ξ) {the (nhop (rt ξ) (dip ξ))}) )]]
    unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(the (dhops (rt ξ) (dip ξ)), dip ξ,
      sqn (rt ξ) (dip ξ), oip ξ, ip ξ)).
    AODV()
    ▷
    [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
      then Some (inc (sqn (rt ξ) rip)) else None) )]]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
    [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
    [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
      then (dests ξ) rip else None) )]]
    groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
    ⊕ ⟨ξ. dip ξ ∉ vD (rt ξ) ∨ sqn (rt ξ) (dip ξ) < dsn ξ ∨ sqnf (rt ξ) (dip ξ) = unk⟩
    broadcast(λξ. rreq(hops ξ + 1, rreqid ξ, dip ξ, max (sqn (rt ξ) (dip ξ)) (dsn ξ),
      dsk ξ, oip ξ, osn ξ, ip ξ)).
    AODV()
  )
  )"

```

| "Γ<sub>AODV</sub> PRrep = labelled PRrep (

```

[[ξ. ξ (| rt := update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, { }) )]]
(
  ⟨ξ. oip ξ = ip ξ⟩
  AODV()
  ⊕ ⟨ξ. oip ξ ≠ ip ξ⟩
  (
    ⟨ξ. oip ξ ∈ vD (rt ξ) ∧ dip ξ ∈ vD (rt ξ)⟩
    [[ξ. ξ (| rt := the (addpreRT (rt ξ) (dip ξ)
      {the (nhop (rt ξ) (oip ξ))}) )]]
    [[ξ. ξ (| rt := the (addpreRT (rt ξ) (the (nhop (rt ξ) (dip ξ))) {the (nhop (rt ξ) (oip ξ))})
    unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(the (dhops (rt ξ) (dip ξ)), dip ξ,
      sqn (rt ξ) (dip ξ), oip ξ, ip ξ)).
    AODV()
    ▷
    [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
      then Some (inc (sqn (rt ξ) rip)) else None) )]]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
    [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
    [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
      then (dests ξ) rip else None) )]]
    groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
    ⊕ ⟨ξ. oip ξ ∉ vD (rt ξ) ∨ dip ξ ∉ vD (rt ξ)⟩
    AODV()
  )
  )

```

```

)
)"

| "ΓAODV PRerr = labelled PRerr (
  [| ξ. ξ (| dests := (λrip. case (dests ξ) rip of None ⇒ None
    | Some rsn ⇒ if rip ∈ vD (rt ξ) ∧ the (nhop (rt ξ) rip) = sip ξ
      ∧ sqn (rt ξ) rip < rsn then Some rsn else None) |])
  [| ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]
  [| ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]
  [| ξ. ξ (| pre := ⋃{ the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } |)]
  [| ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ {})
    then (dests ξ) rip else None) |)]
  groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)). AODV())"

declare ΓAODV.simps [simp del, code del]
lemmas ΓAODV.simps [simp, code] = ΓAODV.simps [simplified]

fun ΓAODV.skeleton
where
  "ΓAODV.skeleton PAadv = seqp_skeleton (ΓAODV PAadv)"
  | "ΓAODV.skeleton PNewPkt = seqp_skeleton (ΓAODV PNewPkt)"
  | "ΓAODV.skeleton PPkt = seqp_skeleton (ΓAODV PPkt)"
  | "ΓAODV.skeleton PRreq = seqp_skeleton (ΓAODV PRreq)"
  | "ΓAODV.skeleton PRrep = seqp_skeleton (ΓAODV PRrep)"
  | "ΓAODV.skeleton PRerr = seqp_skeleton (ΓAODV PRerr)"

lemma ΓAODV.skeleton_wf [simp]:
  "wellformed ΓAODV.skeleton"
proof (rule, intro allI)
  fix pn pn'
  show "call(pn') ∉ stermsl (ΓAODV.skeleton pn)"
  by (cases pn) simp_all
qed

declare ΓAODV.skeleton.simps [simp del, code del]
lemmas ΓAODV.skeleton.simps [simp, code]
  = ΓAODV.skeleton.simps [simplified ΓAODV.simps seqp_skeleton.simps]

lemma aodv_proc_cases [dest]:
  fixes p pn
  shows "p ∈ ctermsl (ΓAODV pn) ⇒
    (p ∈ ctermsl (ΓAODV PAadv) ∨
     p ∈ ctermsl (ΓAODV PNewPkt) ∨
     p ∈ ctermsl (ΓAODV PPkt) ∨
     p ∈ ctermsl (ΓAODV PRreq) ∨
     p ∈ ctermsl (ΓAODV PRrep) ∨
     p ∈ ctermsl (ΓAODV PRerr))"

using assms
by (cases pn) simp_all

definition σAODV :: "ip ⇒ (state × (state, msg, pseqp, pseqp label) seqp) set"
where "σAODV i ≡ {(aodv_init i, ΓAODV PAadv)}"

abbreviation paodv
  :: "ip ⇒ (state × (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
where
  "paodv i ≡ (| init = σAODV i, trans = seqp_sos ΓAODV |)"

lemma aodv_trans: "trans (paodv i) = seqp_sos ΓAODV"
  by simp

lemma aodv_control_within [simp]: "control_within ΓAODV (init (paodv i))"
  unfolding σAODV_def by (rule control_withinI) (auto simp del: ΓAODV.simps)

```

```

lemma aodv_wf [simp]:
  "wellformed  $\Gamma_{AODV}$ "
proof (rule, intro allI)
  fix pn pn'
  show "call(pn')  $\notin$  sterms1 ( $\Gamma_{AODV}$  pn)"
  by (cases pn) simp_all
qed

lemmas aodv_labels_not_empty [simp] = labels_not_empty [OF aodv_wf]

lemma aodv_ex_label [intro]: " $\exists l. l \in \text{labels } \Gamma_{AODV} p$ "
  by (metis aodv_labels_not_empty all_not_in_conv)

lemma aodv_ex_labelE [elim]:
  assumes " $\forall l \in \text{labels } \Gamma_{AODV} p. P l p$ "
  and " $\exists p l. P l p \implies Q$ "
  shows "Q"
  using assms by (metis aodv_ex_label)

lemma aodv_simple_labels [simp]: "simple_labels  $\Gamma_{AODV}$ "
proof
  fix pn p
  assume "p  $\in$  subterms( $\Gamma_{AODV}$  pn)"
  thus " $\exists ! l. \text{labels } \Gamma_{AODV} p = \{l\}$ "
  by (cases pn) (simp_all cong: seqp_congs | elim disjE)+
qed

lemma  $\sigma_{AODV}$ _labels [simp]: " $(\xi, p) \in \sigma_{AODV} i \implies \text{labels } \Gamma_{AODV} p = \{PAodv-:0\}$ "
  unfolding  $\sigma_{AODV}$ _def by simp

lemma aodv_init_kD_empty [simp]:
  " $(\xi, p) \in \sigma_{AODV} i \implies kD (rt \xi) = \{\}$ "
  unfolding  $\sigma_{AODV}$ _def kD_def by simp

lemma aodv_init_sip_not_ip [simp]: " $\neg(\text{sip (aodv_init } i) = i)$ " by simp

lemma aodv_init_sip_not_ip' [simp]:
  assumes " $(\xi, p) \in \sigma_{AODV} i$ "
  shows "sip  $\xi \neq ip \xi$ "
  using assms unfolding  $\sigma_{AODV}$ _def by simp

lemma aodv_init_sip_not_i [simp]:
  assumes " $(\xi, p) \in \sigma_{AODV} i$ "
  shows "sip  $\xi \neq i$ "
  using assms unfolding  $\sigma_{AODV}$ _def by simp

lemma clear_locals_sip_not_ip':
  assumes "ip  $\xi = i$ "
  shows " $\neg(\text{sip (clear_locals } \xi) = i)$ "
  using assms by auto

Stop the simplifier from descending into process terms.
declare seqp_congs [cong]

Configure the main invariant tactic for AODV.
declare
   $\Gamma_{AODV}$ _simps [cterms_env]
  aodv_proc_cases [cterms_cases]
  seq_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans,
    cterms_intros]
  seq_step_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans,
    cterms_intros]

```

end

## 2.4 Invariant assumptions and properties

```
theory B_Aodv_Predicates
imports B_Aodv
begin
```

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

```
abbreviation not_Pkt :: "msg  $\Rightarrow$  bool"
where "not_Pkt m  $\equiv$  case m of Pkt _ _ _  $\Rightarrow$  False | _  $\Rightarrow$  True"
```

```
definition msg_sender :: "msg  $\Rightarrow$  ipc"
where "msg_sender m  $\equiv$  case m of Rreq _ _ _ _ _ ipc  $\Rightarrow$  ipc
      | Rrep _ _ _ _ ipc  $\Rightarrow$  ipc
      | Rerr _ ipc  $\Rightarrow$  ipc
      | Pkt _ _ ipc  $\Rightarrow$  ipc"
```

```
lemma msg_sender_simps [simp]:
  " $\wedge$ hops rreqid dip dsn dsk oip osn sip.
    msg_sender (Rreq hops rreqid dip dsn dsk oip osn sip) = sip"
  " $\wedge$ hops dip dsn oip sip. msg_sender (Rrep hops dip dsn oip sip) = sip"
  " $\wedge$ dests sip. msg_sender (Rerr dests sip) = sip"
  " $\wedge$ d dip sip. msg_sender (Pkt d dip sip) = sip"
unfolding msg_sender_def by simp_all
```

```
definition msg_zhops :: "msg  $\Rightarrow$  bool"
where "msg_zhops m  $\equiv$  case m of
      Rreq hopsc _ dipc _ _ oipc _ sipc  $\Rightarrow$  hopsc = 0  $\longrightarrow$  oipc = sipc
      | Rrep hopsc dipc _ _ sipc  $\Rightarrow$  hopsc = 0  $\longrightarrow$  dipc = sipc
      | _  $\Rightarrow$  True"
```

```
lemma msg_zhops_simps [simp]:
  " $\wedge$ hops rreqid dip dsn dsk oip osn sip.
    msg_zhops (Rreq hops rreqid dip dsn dsk oip osn sip) = (hops = 0  $\longrightarrow$  oip = sip)"
  " $\wedge$ hops dip dsn oip sip. msg_zhops (Rrep hops dip dsn oip sip) = (hops = 0  $\longrightarrow$  dip = sip)"
  " $\wedge$ dests sip. msg_zhops (Rerr dests sip) = True"
  " $\wedge$ d dip. msg_zhops (Newpkt d dip) = True"
  " $\wedge$ d dip sip. msg_zhops (Pkt d dip sip) = True"
unfolding msg_zhops_def by simp_all
```

```
definition rreq_rrep_sn :: "msg  $\Rightarrow$  bool"
where "rreq_rrep_sn m  $\equiv$  case m of Rreq _ _ _ _ _ osnc _  $\Rightarrow$  osnc  $\geq$  1
      | Rrep _ _ dsnc _ _  $\Rightarrow$  dsnc  $\geq$  1
      | _  $\Rightarrow$  True"
```

```
lemma rreq_rrep_sn_simps [simp]:
  " $\wedge$ hops rreqid dip dsn dsk oip osn sip.
    rreq_rrep_sn (Rreq hops rreqid dip dsn dsk oip osn sip) = (osn  $\geq$  1)"
  " $\wedge$ hops dip dsn oip sip. rreq_rrep_sn (Rrep hops dip dsn oip sip) = (dsn  $\geq$  1)"
  " $\wedge$ dests sip. rreq_rrep_sn (Rerr dests sip) = True"
  " $\wedge$ d dip. rreq_rrep_sn (Newpkt d dip) = True"
  " $\wedge$ d dip sip. rreq_rrep_sn (Pkt d dip sip) = True"
unfolding rreq_rrep_sn_def by simp_all
```

```
definition rreq_rrep_fresh :: "rt  $\Rightarrow$  msg  $\Rightarrow$  bool"
where "rreq_rrep_fresh crt m  $\equiv$  case m of Rreq hopsc _ _ _ oipc osnc ipcc  $\Rightarrow$  (ipcc  $\neq$  oipc  $\longrightarrow$ 
      oipc  $\in$  kD(crt)  $\wedge$  (sqn crt oipc > osnc
         $\vee$  (sqn crt oipc = osnc
           $\wedge$  the (dhops crt oipc)  $\leq$  hopsc
           $\wedge$  the (flag crt oipc) = val)))
      | Rrep hopsc dipc dsnc _ ipcc  $\Rightarrow$  (ipcc  $\neq$  dipc  $\longrightarrow$ 
      dipc  $\in$  kD(crt)
       $\wedge$  sqn crt dipc = dsnc
       $\wedge$  the (dhops crt dipc) = hopsc
       $\wedge$  the (flag crt dipc) = val)
```

| \_  $\Rightarrow$  True"

lemma rreq\_rrep\_fresh [simp]:

```
" $\wedge$ hops rreqid dip dsn dsk oip osn sip.
  rreq_rrep_fresh crt (Rreq hops rreqid dip dsn dsk oip osn sip) =
    (sip  $\neq$  oip  $\longrightarrow$  oip $\in$ kD(crt)
      $\wedge$  (sqn crt oip > osn
         $\vee$  (sqn crt oip = osn
             $\wedge$  the (dhops crt oip)  $\leq$  hops
             $\wedge$  the (flag crt oip) = val)))"
" $\wedge$ hops dip dsn oip sip. rreq_rrep_fresh crt (Rrep hops dip dsn oip sip) =
  (sip  $\neq$  dip  $\longrightarrow$  dip $\in$ kD(crt)
    $\wedge$  sqn crt dip = dsn
    $\wedge$  the (dhops crt dip) = hops
    $\wedge$  the (flag crt dip) = val)"
" $\wedge$ dests sip.      rreq_rrep_fresh crt (Rerr dests sip) = True"
" $\wedge$ d dip.         rreq_rrep_fresh crt (Newpkt d dip)   = True"
" $\wedge$ dip sip.      rreq_rrep_fresh crt (Pkt d dip sip)   = True"
unfolding rreq_rrep_fresh_def by simp_all
```

definition rerr\_invalid :: "rt  $\Rightarrow$  msg  $\Rightarrow$  bool"

```
where "rerr_invalid crt m  $\equiv$  case m of Rerr destsc _  $\Rightarrow$  ( $\forall$ rip $\in$ dom(destsc).
  (rip $\in$ iD(crt)  $\wedge$  the (destsc ripc) = sqn crt ripc))
  | _  $\Rightarrow$  True"
```

lemma rerr\_invalid [simp]:

```
" $\wedge$ hops rreqid dip dsn dsk oip osn sip.
  rerr_invalid crt (Rreq hops rreqid dip dsn dsk oip osn sip) = True"
" $\wedge$ hops dip dsn oip sip. rerr_invalid crt (Rrep hops dip dsn oip sip) = True"
" $\wedge$ dests sip.          rerr_invalid crt (Rerr dests sip) = ( $\forall$ rip $\in$ dom(dests).
  rip $\in$ iD(crt)  $\wedge$  the (dests rip) = sqn crt rip)"
" $\wedge$ d dip.             rerr_invalid crt (Newpkt d dip)   = True"
" $\wedge$ dip sip.          rerr_invalid crt (Pkt d dip sip)   = True"
unfolding rerr_invalid_def by simp_all
```

definition

```
initmissing :: "(nat  $\Rightarrow$  state option)  $\times$  'a  $\Rightarrow$  (nat  $\Rightarrow$  state)  $\times$  'a"
```

where

```
"initmissing  $\sigma$  = ( $\lambda$ i. case (fst  $\sigma$ ) i of None  $\Rightarrow$  aadv_init i | Some s  $\Rightarrow$  s, snd  $\sigma$ )"
```

lemma not\_in\_net\_ips\_fst\_init\_missing [simp]:

```
assumes "i  $\notin$  net_ips  $\sigma$ "
  shows "fst (initmissing (netgmap fst  $\sigma$ )) i = aadv_init i"
using assms unfolding initmissing_def by simp
```

lemma fst\_initmissing\_netgmap\_pair\_fst [simp]:

```
"fst (initmissing (netgmap ( $\lambda$ (p, q). (fst (id p), snd (id p), q)) s))
  = fst (initmissing (netgmap fst s))"
unfolding initmissing_def by auto
```

We introduce a streamlined alternative to `initmissing` with `netgmap` to simplify invariant statements and thus facilitate their comprehension and presentation.

lemma fst\_initmissing\_netgmap\_default\_aadv\_init\_netlift:

```
"fst (initmissing (netgmap fst s)) = default aadv_init (netlift fst s)"
unfolding initmissing_def default_def
by (simp add: fst_netgmap_netlift del: One_nat_def)
```

definition

```
netglobal :: "(nat  $\Rightarrow$  state)  $\Rightarrow$  bool  $\Rightarrow$  ((state  $\times$  'b)  $\times$  'c) net_state  $\Rightarrow$  bool"
```

where

```
"netglobal P  $\equiv$  ( $\lambda$ s. P (default aadv_init (netlift fst s)))"
```

end

## 2.5 Quality relations between routes

```
theory B_Fresher
imports B_Aodv_Data
begin
```

### 2.5.1 Net sequence numbers

#### On individual routes

##### definition

```
nsqnr :: "r ⇒ sqn"
```

##### where

```
"nsqnr r ≡ if π4(r) = val ∨ π2(r) = 0 then π2(r) else (π2(r) - 1)"
```

##### lemma nsqnr\_def':

```
"nsqnr r = (if π4(r) = inv then π2(r) - 1 else π2(r))"
```

```
unfolding nsqnr_def by simp
```

##### lemma nsqn<sub>r</sub>\_zero [simp]:

```
"∧ dsn dsk flag hops nhip pre. nsqnr (0, dsk, flag, hops, nhip, pre) = 0"
```

```
unfolding nsqnr_def by clarsimp
```

##### lemma nsqn<sub>r</sub>\_val [simp]:

```
"∧ dsn dsk hops nhip pre. nsqnr (dsn, dsk, val, hops, nhip, pre) = dsn"
```

```
unfolding nsqnr_def by clarsimp
```

##### lemma nsqn<sub>r</sub>\_inv [simp]:

```
"∧ dsn dsk hops nhip pre. nsqnr (dsn, dsk, inv, hops, nhip, pre) = dsn - 1"
```

```
unfolding nsqnr_def by clarsimp
```

##### lemma nsqn<sub>r</sub>\_lte\_dsn [simp]:

```
"∧ dsn dsk flag hops nhip pre. nsqnr (dsn, dsk, flag, hops, nhip, pre) ≤ dsn"
```

```
unfolding nsqnr_def by clarsimp
```

#### On routes in routing tables

##### definition

```
nsqn :: "rt ⇒ ip ⇒ sqn"
```

##### where

```
"nsqn ≡ λrt dip. case σroute(rt, dip) of None ⇒ 0 | Some r ⇒ nsqnr(r)"
```

##### lemma nsqn\_sqn\_def:

```
"∧rt dip. nsqn rt dip = (if flag rt dip = Some val ∨ sqn rt dip = 0
                        then sqn rt dip else sqn rt dip - 1)"
```

```
unfolding nsqn_def sqn_def by (clarsimp split: option.split) auto
```

##### lemma not\_in\_kD\_nsqn [simp]:

```
assumes "dip ∉ kD(rt)"
```

```
shows "nsqn rt dip = 0"
```

```
using assms unfolding nsqn_def by simp
```

##### lemma kD\_nsqn:

```
assumes "dip ∈ kD(rt)"
```

```
shows "nsqn rt dip = nsqnr(the (σroute(rt, dip)))"
```

```
using assms [THEN kD_Some] unfolding nsqn_def by clarsimp
```

##### lemma nsqnr\_r\_flag\_pred [simp, intro]:

```
fixes dsn dsk flag hops nhip pre
```

```
assumes "P (nsqnr (dsn, dsk, val, hops, nhip, pre))"
```

```
and "P (nsqnr (dsn, dsk, inv, hops, nhip, pre))"
```

```
shows "P (nsqnr (dsn, dsk, flag, hops, nhip, pre))"
```

```
using assms by (cases flag) auto
```

##### lemma nsqn<sub>r</sub>\_addpreRT\_inv [simp]:



```

"∧rt dip npre dip'. dip ∈ kD(rt) ⇒
  nsqnr (the (the (addpreRT rt dip npre) dip')) = nsqnr (the (rt dip'))"
unfolding addpreRT_def nsqnr_def
by (frule kD_Some) (clarsimp split: option.split)

lemma sqn_nsqn:
"∧rt dip. sqn rt dip - 1 ≤ nsqn rt dip"
unfolding sqn_def nsqn_def by (clarsimp split: option.split)

lemma nsqn_sqn: "nsqn rt dip ≤ sqn rt dip"
unfolding sqn_def nsqn_def by (cases "rt dip") auto

lemma val_nsqn_sqn [elim, simp]:
  assumes "ip ∈ kD(rt)"
    and "the (flag rt ip) = val"
  shows "nsqn rt ip = sqn rt ip"
using assms unfolding nsqn_sqn_def by auto

lemma vD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ vD(rt)"
  shows "nsqn rt ip = sqn rt ip"
proof -
  from ⟨ip ∈ vD(rt)⟩ have "ip ∈ kD(rt)"
    and "the (flag rt ip) = val" by auto
  thus ?thesis ..
qed

lemma inv_nsqn_sqn [elim, simp]:
  assumes "ip ∈ kD(rt)"
    and "the (flag rt ip) = inv"
  shows "nsqn rt ip = sqn rt ip - 1"
using assms unfolding nsqn_sqn_def by auto

lemma iD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ iD(rt)"
  shows "nsqn rt ip = sqn rt ip - 1"
proof -
  from ⟨ip ∈ iD(rt)⟩ have "ip ∈ kD(rt)"
    and "the (flag rt ip) = inv" by auto
  thus ?thesis ..
qed

lemma nsqn_update_changed_kno_val [simp]: "∧rt ip dsn dsk hops nhip.
  rt ≠ update rt ip (dsn, kno, val, hops, nhip, {})
  ⇒ nsqn (update rt ip (dsn, kno, val, hops, nhip, {})) ip = dsn"
using assms unfolding nsqnr_def update_def
by (clarsimp simp: kD_nsqn split: option.split_asm option.split split_if_asm)
(metis fun_upd_triv)

lemma nsqn_addpreRT_inv [simp]:
"∧rt dip npre dip'. dip ∈ kD(rt) ⇒
  nsqn (the (addpreRT rt dip npre)) dip' = nsqn rt dip'"
unfolding addpreRT_def nsqn_def nsqnr_def
by (frule kD_Some) (clarsimp split: option.split)

lemma nsqn_update_other [simp]:
  fixes dsn dsk flag hops dip nhip pre rt ip
  assumes "dip ≠ ip"
  shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip, pre)) dip = nsqn rt dip"
using assms unfolding nsqn_def
by (clarsimp split: option.split)

lemma nsqn_invalidate_eq:
  assumes "dip ∈ kD(rt)"

```

```

    and "dests dip = Some rsn"
  shows "nsqn (invalidate rt dests) dip = rsn - 1"
using assms
proof -
  from assms obtain dsk hops nhip pre
  where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip, pre)"
  unfolding invalidate_def
  by auto
  moreover from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp
  ultimately show ?thesis
  using ⟨dests dip = Some rsn⟩ by simp
qed

```

```

lemma nsqn_invalidate_other [simp]:
  assumes "dip ∈ kD(rt)"
  and "dip ∉ dom dests"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
using assms by (clarsimp simp add: kD_nsqn)

```

## 2.5.2 Comparing routes

definition

```

fresher :: "r ⇒ r ⇒ bool" ("(_/ ⊆ _)" [51, 51] 50)

```

where

```

"fresher r r' ≡ ((nsqnr r < nsqnr r') ∨ (nsqnr r = nsqnr r' ∧ π5(r) ≥ π5(r')))"

```

```

lemma fresherI1 [intro]:
  assumes "nsqnr r < nsqnr r'"
  shows "r ⊆ r'"
  unfolding fresher_def using assms by simp

```

```

lemma fresherI2 [intro]:
  assumes "nsqnr r = nsqnr r'"
  and "π5(r) ≥ π5(r')"
  shows "r ⊆ r'"
  unfolding fresher_def using assms by simp

```

```

lemma fresherI [intro]:
  assumes "(nsqnr r < nsqnr r') ∨ (nsqnr r = nsqnr r' ∧ π5(r) ≥ π5(r'))"
  shows "r ⊆ r'"
  unfolding fresher_def using assms .

```

```

lemma fresherE [elim]:
  assumes "r ⊆ r'"
  and "nsqnr r < nsqnr r' ⇒ P r r'"
  and "nsqnr r = nsqnr r' ∧ π5(r) ≥ π5(r') ⇒ P r r'"
  shows "P r r'"
  using assms unfolding fresher_def by auto

```

```

lemma fresher_refl [simp]: "r ⊆ r"
  unfolding fresher_def by simp

```

```

lemma fresher_trans [elim, trans]:
  "[ x ⊆ y; y ⊆ z ] ⇒ x ⊆ z"
  unfolding fresher_def by auto

```

```

lemma not_fresher_trans [elim, trans]:
  "[ ¬(x ⊆ y); ¬(z ⊆ x) ] ⇒ ¬(z ⊆ y)"
  unfolding fresher_def by auto

```

```

lemma fresher_dsn_flag_hops_const [simp]:
  fixes dsn dsk dsk' flag hops nhip nhip' pre pre'
  shows "(dsn, dsk, flag, hops, nhip, pre) ⊆ (dsn, dsk', flag, hops, nhip', pre')"
  unfolding fresher_def by (cases flag) simp_all

```

lemma addpre\_fresher [simp]: " $\bigwedge r \text{ npre. } r \sqsubseteq (\text{addpre } r \text{ npre})$ "  
 by clarsimp

### 2.5.3 Comparing routing tables

definition

rt\_fresher :: "ip  $\Rightarrow$  rt  $\Rightarrow$  rt  $\Rightarrow$  bool"

where

"rt\_fresher  $\equiv$   $\lambda$ dip rt rt'. (the ( $\sigma_{\text{route}}(\text{rt}, \text{dip})$ ))  $\sqsubseteq$  (the ( $\sigma_{\text{route}}(\text{rt}', \text{dip})$ ))"

abbreviation

rt\_fresher\_syn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  rt  $\Rightarrow$  bool" (" $\_ / \sqsubseteq \_$ " [51, 999, 51] 50)

where

"rt1  $\sqsubseteq_i$  rt2  $\equiv$  rt\_fresher i rt1 rt2"

lemma rt\_fresher\_def':

"(rt1  $\sqsubseteq_i$  rt2) = (nsqn<sub>r</sub> (the (rt1 i)) < nsqn<sub>r</sub> (the (rt2 i)))  $\vee$   
 nsqn<sub>r</sub> (the (rt1 i)) = nsqn<sub>r</sub> (the (rt2 i))  $\wedge$   $\pi_5$  (the (rt2 i))  $\leq$   $\pi_5$  (the (rt1 i)))"

unfolding rt\_fresher\_def fresher\_def by (rule refl)

lemma single\_rt\_fresher [intro]:

assumes "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"

shows "rt1  $\sqsubseteq_{\text{ip}}$  rt2"

using assms unfolding rt\_fresher\_def .

lemma rt\_fresher\_single [intro]:

assumes "rt1  $\sqsubseteq_{\text{ip}}$  rt2"

shows "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"

using assms unfolding rt\_fresher\_def .

lemma rt\_fresher\_def2:

assumes "dip  $\in$  kD(rt1)"

and "dip  $\in$  kD(rt2)"

shows "(rt1  $\sqsubseteq_{\text{dip}}$  rt2) = (nsqn rt1 dip < nsqn rt2 dip  
 $\vee$  (nsqn rt1 dip = nsqn rt2 dip  
 $\wedge$  the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip)))"

using assms unfolding rt\_fresher\_def fresher\_def by (simp add: kD\_nsqn\_proj5\_eq\_dhops)

lemma rt\_fresherI1 [intro]:

assumes "dip  $\in$  kD(rt1)"

and "dip  $\in$  kD(rt2)"

and "nsqn rt1 dip < nsqn rt2 dip"

shows "rt1  $\sqsubseteq_{\text{dip}}$  rt2"

unfolding rt\_fresher\_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt\_fresherI2 [intro]:

assumes "dip  $\in$  kD(rt1)"

and "dip  $\in$  kD(rt2)"

and "nsqn rt1 dip = nsqn rt2 dip"

and "the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip)"

shows "rt1  $\sqsubseteq_{\text{dip}}$  rt2"

unfolding rt\_fresher\_def2 [OF assms(1-2)] using assms(3-4) by simp

lemma rt\_fresherE [elim]:

assumes "rt1  $\sqsubseteq_{\text{dip}}$  rt2"

and "dip  $\in$  kD(rt1)"

and "dip  $\in$  kD(rt2)"

and "[[ nsqn rt1 dip < nsqn rt2 dip ]  $\implies$  P rt1 rt2 dip]"

and "[[ nsqn rt1 dip = nsqn rt2 dip;  
 the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip) ]  $\implies$  P rt1 rt2 dip]"

shows "P rt1 rt2 dip"

using assms(1) unfolding rt\_fresher\_def2 [OF assms(2-3)]

using assms(4-5) by auto

```

lemma rt_fresher_refl [simp]: "rt  $\sqsubseteq_{dip}$  rt"
  unfolding rt_fresher_def by simp

lemma rt_fresher_trans [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
  using assms unfolding rt_fresher_def by auto

lemma rt_fresher_if_Some [intro!]:
  assumes "the (rt dip)  $\sqsubseteq$  r"
  shows "rt  $\sqsubseteq_{dip}$  ( $\lambda ip. \text{if } ip = \text{dip then Some } r \text{ else rt } ip$ )"
  using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip  $\Rightarrow$  rt  $\Rightarrow$  rt  $\Rightarrow$  bool"
where
  "rt_fresh_as  $\equiv$   $\lambda dip \text{ } rt1 \text{ } rt2. (rt1 \sqsubseteq_{dip} rt2) \wedge (rt2 \sqsubseteq_{dip} rt1)$ "

abbreviation
  rt_fresh_as_syn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  rt  $\Rightarrow$  bool" ("(_/  $\approx_{dip}$  _)" [51, 999, 51] 50)
where
  "rt1  $\approx_i$  rt2  $\equiv$  rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_refl [simp]: " $\wedge rt \text{ dip}. rt \approx_{dip} rt$ "
  unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
  " $\wedge rt1 \text{ } rt2 \text{ } rt3 \text{ } dip. [rt1 \approx_{dip} rt2; rt2 \approx_{dip} rt3] \implies rt1 \approx_{dip} rt3$ "
  unfolding rt_fresh_as_def rt_fresher_def
  by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt1"
  shows "rt1  $\approx_{dip}$  rt2"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
  assumes "dip  $\in$  kD(rt)"
    and "dip  $\in$  kD(rt')"
    and "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"
    and "the (rt2 dip)  $\sqsubseteq$  the (rt1 dip)"
  shows "rt1  $\approx_{dip}$  rt2"
  using assms unfolding rt_fresh_as_def
  by (clarsimp dest!: single_rt_fresher)

lemma nsqn_rt_fresh_asI:
  assumes "dip  $\in$  kD(rt)"
    and "dip  $\in$  kD(rt')"
    and "nsqn rt dip = nsqn rt' dip"
    and " $\pi_5(\text{the } (rt \text{ dip})) = \pi_5(\text{the } (rt' \text{ dip}))$ "
  shows "rt  $\approx_{dip}$  rt'"
proof
  from assms(1-2,4) have dhops': "the (dhops rt' dip)  $\leq$  the (dhops rt dip)"
  by (simp add: proj5_eq_dhops)
  with assms(1-3) show "rt  $\sqsubseteq_{dip}$  rt'"
  by (rule rt_fresherI2)
next
  from assms(1-2,4) have dhops: "the (dhops rt dip)  $\leq$  the (dhops rt' dip)"
  by (simp add: proj5_eq_dhops)
  with assms(2,1) assms(3) [symmetric] show "rt'  $\sqsubseteq_{dip}$  rt"
  by (rule rt_fresherI2)
qed

```

```

lemma rt_fresh_asE [elim]:
  assumes "rt1  $\approx_{dip}$  rt2"
    and "[[ rt1  $\sqsubseteq_{dip}$  rt2; rt2  $\sqsubseteq_{dip}$  rt1 ] ]  $\implies$  P rt1 rt2 dip"
  shows "P rt1 rt2 dip"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD1 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\sqsubseteq_{dip}$  rt1"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\approx_{dip}$  rt1"
  using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
  assumes " $\neg$  (rt1  $\sqsubseteq_{dip}$  rt2)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
  proof
    assume "rt1  $\approx_{dip}$  rt2"
    hence "rt1  $\sqsubseteq_{dip}$  rt2" ..
    with ( $\neg$  (rt1  $\sqsubseteq_{dip}$  rt2)) show False ..
  qed

lemma not_rt_fresh_asI2 [intro]:
  assumes " $\neg$  (rt2  $\sqsubseteq_{dip}$  rt1)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
  proof
    assume "rt1  $\approx_{dip}$  rt2"
    hence "rt2  $\sqsubseteq_{dip}$  rt1" ..
    with ( $\neg$  (rt2  $\sqsubseteq_{dip}$  rt1)) show False ..
  qed

lemma not_single_rt_fresher [elim]:
  assumes " $\neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))"
  shows " $\neg$ (rt1  $\sqsubseteq_{ip}$  rt2)"
  proof
    assume "rt1  $\sqsubseteq_{ip}$  rt2"
    hence "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)" ..
    with ( $\neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))) show False ..
  qed

lemmas not_single_rt_fresh_asI1 [intro] = not_rt_fresh_asI1 [OF not_single_rt_fresher]
lemmas not_single_rt_fresh_asI2 [intro] = not_rt_fresh_asI2 [OF not_single_rt_fresher]

lemma not_rt_fresher_single [elim]:
  assumes " $\neg$ (rt1  $\sqsubseteq_{ip}$  rt2)"
  shows " $\neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))"
  proof
    assume "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"
    hence "rt1  $\sqsubseteq_{ip}$  rt2" ..
    with ( $\neg$ (rt1  $\sqsubseteq_{ip}$  rt2)) show False ..
  qed

lemma rt_fresh_as_nsqr:
  assumes "dip  $\in$  kD(rt1)"
    and "dip  $\in$  kD(rt2)"

```

```

    and "rt1  $\approx_{dip}$  rt2"
    shows "nsqnr (the (rt2 dip)) = nsqnr (the (rt1 dip))"
using assms(3) unfolding rt_fresher_as_def
by (auto simp: rt_fresher_def2 [OF ⟨dip ∈ kD(rt1)⟩ ⟨dip ∈ kD(rt2)⟩]
    rt_fresher_def2 [OF ⟨dip ∈ kD(rt2)⟩ ⟨dip ∈ kD(rt1)⟩]
    kD_nsqn [OF ⟨dip ∈ kD(rt1)⟩]
    kD_nsqn [OF ⟨dip ∈ kD(rt2)⟩])

lemma rt_fresher_mapupd [intro!]:
  assumes "dip ∈ kD(rt)"
    and "the (rt dip)  $\sqsubseteq$  r"
  shows "rt  $\sqsubseteq_{dip}$  rt(dip  $\mapsto$  r)"
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_map_update_other [intro!]:
  assumes "dip ∈ kD(rt)"
    and "dip  $\neq$  ip"
  shows "rt  $\sqsubseteq_{dip}$  rt(ip  $\mapsto$  r)"
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_update_other [simp]:
  assumes inkD: "dip ∈ kD(rt)"
    and "dip  $\neq$  ip"
  shows "rt  $\sqsubseteq_{dip}$  update rt ip r"
using assms unfolding update_def
by (clarsimp split: option.split) (fastforce)

theorem rt_fresher_update [simp]:
  assumes "dip ∈ kD(rt)"
    and "the (dhops rt dip)  $\geq$  1"
    and "update_arg_wf r"
  shows "rt  $\sqsubseteq_{dip}$  update rt ip r"
proof (cases "dip = ip")
  assume "dip  $\neq$  ip" with ⟨dip ∈ kD(rt)⟩ show ?thesis
    by (rule rt_fresher_update_other)
next
  assume "dip = ip"

  from ⟨dip ∈ kD(rt)⟩ obtain dsnn dskn fn hopsn nhipn pren
    where rtn [simp]: "the (rt dip) = (dsnn, dskn, fn, hopsn, nhipn, pren)"
    by (metis prod_cases6)
  with ⟨the (dhops rt dip)  $\geq$  1⟩ and ⟨dip ∈ kD(rt)⟩ have "hopsn  $\geq$  1"
    by (metis proj5_eq_dhops projs(4))
  from ⟨dip ∈ kD(rt)⟩ rtn have [simp]: "sqn rt dip = dsnn"
    and [simp]: "the (dhops rt dip) = hopsn"
    and [simp]: "the (flag rt dip) = fn"
  by (simp add: sqn_def proj5_eq_dhops [symmetric]
    proj4_eq_flag [symmetric])

  from ⟨update_arg_wf r⟩ have "(dsnn, dskn, fn, hopsn, nhipn, pren)
     $\sqsubseteq$  the ((update rt dip r) dip)"
  proof (rule wf_r_cases)
    fix nhip pre
    from ⟨hopsn  $\geq$  1⟩ have " $\wedge$ pre'. (dsnn, dskn, fn, hopsn, nhipn, pren)
       $\sqsubseteq$  (dsnn, unk, val, Suc 0, nhip, pre')"
    unfolding fresher_def sqn_def by (cases fn) auto
    thus "(dsnn, dskn, fn, hopsn, nhipn, pren)
       $\sqsubseteq$  the (update rt dip (0, unk, val, Suc 0, nhip, pre) dip)"
    using ⟨dip ∈ kD(rt)⟩ by - (rule update_cases_kD, simp_all)
  next
    fix dsn :: sqn and hops nhip pre
    assume "0 < dsn"
    show "(dsnn, dskn, fn, hopsn, nhipn, pren)
       $\sqsubseteq$  the (update rt dip (dsn, kno, val, hops, nhip, pre) dip)"

```

```

proof (rule update_cases_kD [OF _ ⟨dip∈kD(rt)⟩], simp_all add: ⟨0 < dsn⟩)
  assume "dsnn < dsn"
  thus "(dsnn, dskn, fn, hopsn, nhipn, pren)
    ⊆ (dsn, kno, val, hops, nhip, pre ∪ pren)"
  unfolding fresher_def by auto
next
  assume "dsnn = dsn"
  and "hops < hopsn"
  thus "(dsn, dskn, fn, hopsn, nhipn, pren)
    ⊆ (dsn, kno, val, hops, nhip, pre ∪ pren)"
  unfolding fresher_def nsqnr_def by simp
next
  assume "dsnn = dsn"
  with ⟨0 < dsn⟩
  show "(dsn, dskn, inv, hopsn, nhipn, pren)
    ⊆ (dsn, kno, val, hops, nhip, pre ∪ pren)"
  unfolding fresher_def by simp
qed
qed
hence "rt ⊆dip update rt dip r"
  by - (rule single_rt_fresher, simp)
with ⟨dip = ip⟩ show ?thesis by simp
qed

```

theorem *rt\_fresher\_invalidate* [simp]:

```

assumes "dip∈kD(rt)"
  and indests: "∀rip∈dom(dests). rip∈vD(rt) ∧ sqn rt rip < the (dests rip)"
shows "rt ⊆dip invalidate rt dests"
proof (cases "dip∈dom(dests)")
  assume "dip∉dom(dests)"
  thus ?thesis using ⟨dip∈kD(rt)⟩
  by - (rule single_rt_fresher, simp)
next
  assume "dip∈dom(dests)"
  moreover with indests have "dip∈vD(rt)"
    and "sqn rt dip < the (dests dip)"
  by auto
ultimately show ?thesis
  unfolding invalidate_def sqn_def
  by - (rule single_rt_fresher, auto simp: fresher_def)
qed

```

lemma *nsqn<sub>r</sub>\_invalidate* [simp]:

```

assumes "dip∈kD(rt)"
  and "dip∈dom(dests)"
shows "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
using assms unfolding invalidate_def by auto

```

lemma *rt\_fresh\_as\_inc\_invalidate* [simp]:

```

assumes "dip∈kD(rt)"
  and "∀rip∈dom(dests). rip∈vD(rt) ∧ the (dests rip) = inc (sqn rt rip)"
shows "rt ≈dip invalidate rt dests"
proof (cases "dip∈dom(dests)")
  assume "dip∉dom(dests)"
  with ⟨dip∈kD(rt)⟩ have "dip∈kD(invalidate rt dests)"
  by simp
  with ⟨dip∈kD(rt)⟩ show ?thesis
  by rule (simp_all add: ⟨dip∉dom(dests)⟩)
next
  assume "dip∈dom(dests)"
  with assms(2) have "dip∈vD(rt)"
    and "the (dests dip) = inc (sqn rt dip)" by auto
  from ⟨dip∈vD(rt)⟩ have "dip∈kD(rt)" by simp
  moreover then have "dip∈kD(invalidate rt dests)" by simp

```

```

ultimately show ?thesis
proof (rule nsqn_rt_fresh_asI)
  from ⟨dip∈vD(rt)⟩ have "nsqn rt dip = sqn rt dip" by simp
  also have "sqn rt dip = nsqnr (the (invalidate rt dests dip))"
  proof -
    from ⟨dip∈kD(rt)⟩ have "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
      using ⟨dip∈dom(dests)⟩ by (rule nsqnr_invalidate)
    with ⟨the (dests dip) = inc (sqn rt dip)⟩
    show "sqn rt dip = nsqnr (the (invalidate rt dests dip))" by simp
  qed
  also from ⟨dip∈kD(invalidate rt dests)⟩
    have "nsqnr (the (invalidate rt dests dip)) = nsqn (invalidate rt dests) dip"
      by (simp add: kD_nsqn)
    finally show "nsqn rt dip = nsqn (invalidate rt dests) dip" .
  qed simp
qed

```

```
lemmas rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]
```

```

lemma rt_fresh_as_addpreRT [simp]:
  assumes "ip∈kD(rt)"
  shows "rt ≈dip the (addpreRT rt ip npre)"
  using assms [THEN kD_Some] by (auto simp: addpreRT_def)

```

```
lemmas rt_fresher_addpreRT [simp] = rt_fresh_as_addpreRT [THEN rt_fresh_asD1]
```

## 2.5.4 Strictly comparing routing tables

```
definition rt_strictly_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
```

where

```
"rt_strictly_fresher ≡ λdip rt1 rt2. (rt1 ⊆dip rt2) ∧ ¬(rt1 ≈dip rt2)"
```

abbreviation

```
rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊆ _)" [51, 999, 51] 50)
```

where

```
"rt1 ⊆i rt2 ≡ rt_strictly_fresher i rt1 rt2"
```

```
lemma rt_strictly_fresher_def'':
```

```
"rt1 ⊆i rt2 = ((rt1 ⊆i rt2) ∧ ¬(rt2 ⊆i rt1))"
unfolding rt_strictly_fresher_def rt_fresh_as_def by auto
```

```
lemma rt_strictly_fresherI' [intro]:
```

```

assumes "rt1 ⊆i rt2"
  and "¬(rt2 ⊆i rt1)"
  shows "rt1 ⊆i rt2"
  using assms unfolding rt_strictly_fresher_def'' by simp

```

```
lemma rt_strictly_fresherE' [elim]:
```

```

assumes "rt1 ⊆i rt2"
  and "[| rt1 ⊆i rt2; ¬(rt2 ⊆i rt1) |] ⇒ P rt1 rt2 i"
  shows "P rt1 rt2 i"
  using assms unfolding rt_strictly_fresher_def'' by simp

```

```
lemma rt_strictly_fresherI [intro]:
```

```

assumes "rt1 ⊆i rt2"
  and "¬(rt1 ≈i rt2)"
  shows "rt1 ⊆i rt2"
  unfolding rt_strictly_fresher_def using assms ..

```

```
lemmas rt_strictly_fresher_singleI [elim] = rt_strictly_fresherI [OF single_rt_fresher]
```

```
lemma rt_strictly_fresherE [elim]:
```

```

assumes "rt1 ⊆i rt2"
  and "[| rt1 ⊆i rt2; ¬(rt1 ≈i rt2) |] ⇒ P rt1 rt2 i"

```



```

  shows "P rt1 rt2 i"
using assms(1) unfolding rt_strictly_fresher_def
by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':
  "rt1  $\sqsubseteq_i$  rt2 =
    (nsqnr (the (rt1 i)) < nsqnr (the (rt2 i))
       $\vee$  (nsqnr (the (rt1 i)) = nsqnr (the (rt2 i))  $\wedge$   $\pi_5$ (the (rt1 i)) >  $\pi_5$ (the (rt2 i))))"
  unfolding rt_strictly_fresher_def'' rt_fresher_def fresher_def by auto

lemma rt_strictly_fresher_fresherD [dest]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  shows "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"
  using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  shows " $\neg$  rt1  $\approx_{dip}$  rt2"
  using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
  using assms proof -
    from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  obtain "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)" by auto
    also from  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  obtain "the (rt2 dip)  $\sqsubseteq$  the (rt3 dip)" by auto
    finally have "the (rt1 dip)  $\sqsubseteq$  the (rt3 dip)" .

    moreover have " $\neg$  (rt1  $\approx_{dip}$  rt3)"
  proof -
    from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  obtain " $\neg$ (the (rt2 dip)  $\sqsubseteq$  the (rt1 dip))" by auto
    also from  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  obtain " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt2 dip))" by auto
    finally have " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt1 dip))" .
    thus ?thesis ..
  qed
  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: " $\neg$  (rt  $\sqsubseteq_{dip}$  rt)"
  using assms unfolding rt_strictly_fresher_def
  by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
  proof -
    from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  have "rt1  $\sqsubseteq_{dip}$  rt2"
    and " $\neg$ (rt2  $\sqsubseteq_{dip}$  rt1)"
    unfolding rt_strictly_fresher_def'' by auto
    from this(1) and  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  have "rt1  $\sqsubseteq_{dip}$  rt3" ..

    moreover from  $\langle$  $\neg$ (rt2  $\sqsubseteq_{dip}$  rt1) $\rangle$  have " $\neg$ (rt3  $\sqsubseteq_{dip}$  rt1)"
  proof (rule contrapos_nn)
    assume "rt3  $\sqsubseteq_{dip}$  rt1"
    with  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  show "rt2  $\sqsubseteq_{dip}$  rt1" ..
  qed

  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3"
  unfolding rt_strictly_fresher_def'' by auto
qed

lemma rt_fresher_trans_rt_strictly_fresher' [elim, trans]:

```

```

assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
proof -
  from ⟨rt2  $\sqsubseteq_{dip}$  rt3⟩ have "rt2  $\sqsubseteq_{dip}$  rt3"
    and "¬(rt3  $\sqsubseteq_{dip}$  rt2)"
  unfolding rt_strictly_fresher_def'' by auto
  from ⟨rt1  $\sqsubseteq_{dip}$  rt2⟩ and this(1) have "rt1  $\sqsubseteq_{dip}$  rt3" ..

  moreover from ⟨¬(rt3  $\sqsubseteq_{dip}$  rt2)⟩ have "¬(rt3  $\sqsubseteq_{dip}$  rt1)"
  proof (rule contrapos_nn)
    assume "rt3  $\sqsubseteq_{dip}$  rt1"
    thus "rt3  $\sqsubseteq_{dip}$  rt2" using ⟨rt1  $\sqsubseteq_{dip}$  rt2⟩ ..
  qed

  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3"
  unfolding rt_strictly_fresher_def'' by auto
qed

lemma rt_fresher_imp_nsqn_le:
  assumes "rt1  $\sqsubseteq_{ip}$  rt2"
    and "ip  $\in$  kD rt1"
    and "ip  $\in$  kD rt2"
  shows "nsqn rt1 ip  $\leq$  nsqn rt2 ip"
using assms(1)
by (auto simp add: rt_fresher_def2 [OF assms(2-3)])

lemma rt_strictly_fresher_ltI [intro]:
  assumes "dip  $\in$  kD(rt1)"
    and "dip  $\in$  kD(rt2)"
    and "nsqn rt1 dip < nsqn rt2 dip"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
proof
  from assms show "rt1  $\sqsubseteq_{dip}$  rt2" ..
next
  show "¬(rt1  $\approx_{dip}$  rt2)"
  proof
    assume "rt1  $\approx_{dip}$  rt2"
    hence "rt2  $\sqsubseteq_{dip}$  rt1" ..
    hence "nsqn rt2 dip  $\leq$  nsqn rt1 dip"
      using ⟨dip  $\in$  kD(rt2)⟩ ⟨dip  $\in$  kD(rt1)⟩
      by (rule rt_fresher_imp_nsqn_le)
    with ⟨nsqn rt1 dip < nsqn rt2 dip⟩ show "False"
      by simp
  qed
qed

lemma rt_strictly_fresher_eqI [intro]:
  assumes "i  $\in$  kD(rt1)"
    and "i  $\in$  kD(rt2)"
    and "nsqn rt1 i = nsqn rt2 i"
    and " $\pi_5(\text{the } (rt2 \ i)) < \pi_5(\text{the } (rt1 \ i))$ "
  shows "rt1  $\sqsubseteq_i$  rt2"
using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

lemma invalidate_rtsf_left [simp]:
  " $\bigwedge$ dests dip rt rt'. dests dip = None  $\implies$  (invalidate rt dests  $\sqsubseteq_{dip}$  rt') = (rt  $\sqsubseteq_{dip}$  rt')"
unfolding invalidate_def rt_strictly_fresher_def'
by (rule iffI) (auto split: option.split_asm)

lemma vD_invalidate_rt_strictly_fresher [simp]:
  assumes "dip  $\in$  vD(invalidate rt1 dests)"
  shows "(invalidate rt1 dests  $\sqsubseteq_{dip}$  rt2) = (rt1  $\sqsubseteq_{dip}$  rt2)"
proof (cases "dip  $\in$  dom(dests)")

```

```

assume "dip ∈ dom(dests)"
hence "dip ∉ vD(invalidate rt1 dests)"
  unfolding invalidate_def vD_def
  by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
with ⟨dip ∈ vD(invalidate rt1 dests)⟩ show ?thesis by simp
next
assume "dip ∉ dom(dests)"
hence "dests dip = None" by auto
moreover with ⟨dip ∈ vD(invalidate rt1 dests)⟩ have "dip ∈ vD(rt1)"
  unfolding invalidate_def vD_def
  by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
ultimately show ?thesis
  unfolding invalidate_def rt_strictly_fresher_def' by auto
qed

lemma rt_strictly_fresher_update_other [elim!]:
  "∧dip ip rt r rt'. [ dip ≠ ip; rt ⊆dip rt' ] ⇒ update rt ip r ⊆dip rt'"
  unfolding rt_strictly_fresher_def' by clarsimp

lemma addpreRT_strictly_fresher [simp]:
  assumes "dip ∈ kD(rt)"
  shows "(the (addpreRT rt dip npre) ⊆ip rt2) = (rt ⊆ip rt2)"
  using assms unfolding rt_strictly_fresher_def' by clarsimp

lemma lt_sqn_imp_update_strictly_fresher:
  assumes "dip ∈ vD (rt2 nhip)"
  and *: "osn < sqn (rt2 nhip) dip"
  and **: "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
  shows "update rt dip (osn, kno, val, hops, nhip, {}) ⊆dip rt2 nhip"
  unfolding rt_strictly_fresher_def'
  proof (rule disjI1)
    from ** have "nsqn (update rt dip (osn, kno, val, hops, nhip, {})) dip = osn"
      by (rule nsqn_update_changed_kno_val)
    with ⟨dip ∈ vD(rt2 nhip)⟩
      have "nsqnr (the (update rt dip (osn, kno, val, hops, nhip, {}) dip)) = osn"
        by (simp add: kD_nsqn)
    also have "osn < sqn (rt2 nhip) dip" by (rule *)
    also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip dip))"
      unfolding nsqnr_def using ⟨dip ∈ vD (rt2 nhip)⟩
      by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
    finally show "nsqnr (the (update rt dip (osn, kno, val, hops, nhip, {}) dip))
      < nsqnr (the (rt2 nhip dip))" .
  qed

lemma dhops_le_hops_imp_update_strictly_fresher:
  assumes "dip ∈ vD(rt2 nhip)"
  and sqn: "sqn (rt2 nhip) dip = osn"
  and hop: "the (dhops (rt2 nhip) dip) ≤ hops"
  and **: "rt ≠ update rt dip (osn, kno, val, Suc hops, nhip, {})"
  shows "update rt dip (osn, kno, val, Suc hops, nhip, {}) ⊆dip rt2 nhip"
  unfolding rt_strictly_fresher_def'
  proof (rule disjI2, rule conjI)
    from ** have "nsqn (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip = osn"
      by (rule nsqn_update_changed_kno_val)
    with ⟨dip ∈ vD(rt2 nhip)⟩
      have "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip)) = osn"
        by (simp add: kD_nsqn)
    also have "osn = sqn (rt2 nhip) dip" by (rule sqn [symmetric])
    also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip dip))"
      unfolding nsqnr_def using ⟨dip ∈ vD(rt2 nhip)⟩
      by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
    finally show "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip))
      = nsqnr (the (rt2 nhip dip))" .
  next

```

```

have "the (dhops (rt2 nhip) dip) ≤ hops" by (rule hop)
also have "hops < hops + 1" by simp
also have "hops + 1 = the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, { })) dip)"
  using ** by simp
finally have "the (dhops (rt2 nhip) dip)
  < the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, { })) dip)" .
thus "π5 (the (rt2 nhip) dip) < π5 (the (update rt dip (osn, kno, val, Suc hops, nhip, { })) dip)"
  using ⟨dip ∈ vD(rt2 nhip)⟩ by (simp add: proj5_eq_dhops)
qed

```

```

lemma nsqn_invalidate:
  assumes "dip ∈ kD(rt)"
    and "∀ip∈dom(dests). ip ∈ vD(rt) ∧ the (dests ip) = inc (sqn rt ip)"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
proof -
  from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp

  from assms have "rt ≈dip invalidate rt dests"
    by (rule rt_fresh_as_inc_invalidate)
  with ⟨dip ∈ kD(rt)⟩ ⟨dip ∈ kD(invalidate rt dests)⟩ show ?thesis
    by (simp add: kD_nsqn del: invalidate_kD_inv)
      (erule(2) rt_fresh_as_nsqnr)
qed

```

end

## 2.6 Invariant proofs on individual processes

```

theory B_Seq_Invariants
imports "../.../AWN/Invariants" B_Aadv B_Aadv_Data B_Aadv_Predicates B_Fresher

```

begin

The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

```

lemma sequence_number_increases:
  "paadv i ⊨A onll ΓAODV (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
  by inv_cterms

```

```

lemma sequence_number_one_or_bigger:
  "paadv i ⊨ onl ΓAODV (λ(ξ, _). 1 ≤ sn ξ)"
  by (rule onll_step_to_invariantI [OF sequence_number_increases])
    (auto simp: σAODV_def)

```

We can get rid of the onl/onll if desired...

```

lemma sequence_number_increases':
  "paadv i ⊨A (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
  by (rule step_invariant_weakenE [OF sequence_number_increases]) (auto dest!: onllD)

```

```

lemma sequence_number_one_or_bigger':
  "paadv i ⊨ (λ(ξ, _). 1 ≤ sn ξ)"
  by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto

```

```

lemma sip_in_kD:
  "paadv i ⊨ onl ΓAODV (λ(ξ, l). l ∈ ({PAadv-:7} ∪ {PAadv-:5} ∪ {PRrep-:0..PRrep-:4}
    ∪ {PRreq-:0..PRreq-:3}) → sip ξ ∈ kD (rt ξ))"
  by inv_cterms

```

```

lemma addpreRT_partly_welldefined:
  "paadv i ⊨
  onl ΓAODV (λ(ξ, l). (l ∈ {PRreq-:16..PRreq-:18} ∪ {PRrep-:1..PRrep-:5} → dip ξ ∈ kD (rt ξ)))"

```

$\wedge (1 \in \{\text{PRreq-:3..PRreq-:17}\} \longrightarrow \text{oip } \xi \in \text{kD } (\text{rt } \xi))"$

by *inv\_cterms*

Proposition 7.38

lemma *includes\_nhip*:

"paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, l). \forall \text{dip} \in \text{kD}(\text{rt } \xi). \text{the } (\text{nhop } (\text{rt } \xi) \text{ dip}) \in \text{kD}(\text{rt } \xi))"$

proof -

{ fix ip and  $\xi \xi' :: \text{state}$

assume " $\forall \text{dip} \in \text{kD} (\text{rt } \xi). \text{the } (\text{nhop } (\text{rt } \xi) \text{ dip}) \in \text{kD} (\text{rt } \xi)"$

and " $\xi' = \xi(\text{rt} := \text{update } (\text{rt } \xi) \text{ ip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{ip}, \{\})"$ )"

hence " $\forall \text{dip} \in \text{kD} (\text{rt } \xi).$

the  $(\text{nhop } (\text{update } (\text{rt } \xi) \text{ ip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{ip}, \{\})) \text{ dip}) = \text{ip}$

$\vee$  the  $(\text{nhop } (\text{update } (\text{rt } \xi) \text{ ip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{ip}, \{\})) \text{ dip}) \in \text{kD} (\text{rt } \xi)"$

by *clarsimp (metis nhop\_update\_unk\_val update\_another)*

} note *one\_hop* = this

{ fix ip sip sn hops and  $\xi \xi' :: \text{state}$

assume " $\forall \text{dip} \in \text{kD} (\text{rt } \xi). \text{the } (\text{nhop } (\text{rt } \xi) \text{ dip}) \in \text{kD} (\text{rt } \xi)"$

and " $\xi' = \xi(\text{rt} := \text{update } (\text{rt } \xi) \text{ ip } (\text{sn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip}, \{\})"$ )"

and " $\text{sip} \in \text{kD} (\text{rt } \xi)"$

hence " $(\text{the } (\text{nhop } (\text{update } (\text{rt } \xi) \text{ ip } (\text{sn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip}, \{\})) \text{ ip}) = \text{ip}$

$\vee$  the  $(\text{nhop } (\text{update } (\text{rt } \xi) \text{ ip } (\text{sn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip}, \{\})) \text{ ip}) \in \text{kD} (\text{rt } \xi)"$

$\wedge (\forall \text{dip} \in \text{kD} (\text{rt } \xi).$

the  $(\text{nhop } (\text{update } (\text{rt } \xi) \text{ ip } (\text{sn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip}, \{\})) \text{ dip}) = \text{ip}$

$\vee$  the  $(\text{nhop } (\text{update } (\text{rt } \xi) \text{ ip } (\text{sn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip}, \{\})) \text{ dip}) \in \text{kD} (\text{rt } \xi)"$ )"

by *(metis kD\_update\_unchanged nhop\_update\_changed update\_another)*

} note *nhip\_is\_sip* = this

show ?thesis

by (*inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf sip\_in\_kD]*

*onl\_invariant\_sterms [OF aadv\_wf addpreRT\_partly\_welldefined]*

*solve: one\_hop nhip\_is\_sip*)

qed

Proposition 7.22: needed in Proposition 7.4

lemma *addpreRT\_welldefined*:

"paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, l). (1 \in \{\text{PRreq-:16..PRreq-:18}\} \longrightarrow \text{dip } \xi \in \text{kD} (\text{rt } \xi)) \wedge$

$(1 = \text{PRreq-:17} \longrightarrow \text{oip } \xi \in \text{kD} (\text{rt } \xi)) \wedge$

$(1 = \text{PRrep-:4} \longrightarrow \text{dip } \xi \in \text{kD} (\text{rt } \xi)) \wedge$

$(1 = \text{PRrep-:5} \longrightarrow (\text{the } (\text{nhop } (\text{rt } \xi) (\text{dip } \xi))) \in \text{kD} (\text{rt } \xi)))"$

(is " $\_ \models$  onl  $\Gamma_{AODV} ?P"$ )

unfolding *invariant\_def*

proof

fix s

assume " $s \in \text{reachable } (\text{paadv } i) \text{ TT}"$

then obtain  $\xi p$  where " $s = (\xi, p)"$

and " $(\xi, p) \in \text{reachable } (\text{paadv } i) \text{ TT}"$

by *(metis PairE)*

have " $\text{onl } \Gamma_{AODV} ?P (\xi, p)"$

proof (rule *onlI*)

fix l

assume " $l \in \text{labels } \Gamma_{AODV} p"$

with  $\langle (\xi, p) \in \text{reachable } (\text{paadv } i) \text{ TT} \rangle$

have *I1*: " $l \in \{\text{PRreq-:16..PRreq-:18}\} \longrightarrow \text{dip } \xi \in \text{kD}(\text{rt } \xi)"$

and *I2*: " $l = \text{PRreq-:17} \longrightarrow \text{oip } \xi \in \text{kD}(\text{rt } \xi)"$

and *I3*: " $l \in \{\text{PRrep-:1..PRrep-:5}\} \longrightarrow \text{dip } \xi \in \text{kD}(\text{rt } \xi)"$

by *(auto dest!: invariantD [OF addpreRT\_partly\_welldefined])*

moreover from  $\langle (\xi, p) \in \text{reachable } (\text{paadv } i) \text{ TT} \rangle$   $\langle l \in \text{labels } \Gamma_{AODV} p \rangle$  and *I3*

have " $l = \text{PRrep-:5} \longrightarrow (\text{the } (\text{nhop } (\text{rt } \xi) (\text{dip } \xi))) \in \text{kD}(\text{rt } \xi)"$

by *(auto dest!: invariantD [OF includes\_nhip])*

ultimately show " $?P (\xi, l)"$

by *simp*

qed

with  $\langle s = (\xi, p) \rangle$  show " $\text{onl } \Gamma_{AODV} ?P s"$

by *simp*

qed

Proposition 7.4

**lemma known\_destinations\_increase:**

```
"paadv i  $\models_A$  onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). kD (rt \xi) \subseteq kD (rt \xi'))$ "
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf addpreRT_welldefined]
    simp add: subset_insertI)
```

Proposition 7.5

**lemma rreqs\_increase:**

```
"paadv i  $\models_A$  onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). rreqs \xi \subseteq rreqs \xi')$ "
by (inv_cterms simp add: subset_insertI)
```

**lemma dests\_bigger\_than\_sqn:**

```
"paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, l). l \in \{PAadv-:15..PAadv-:19\}$ 
     $\cup \{PPkt-:7..PPkt-:11\}$ 
     $\cup \{PRreq-:9..PRreq-:13\}$ 
     $\cup \{PRreq-:21..PRreq-:25\}$ 
     $\cup \{PRrep-:9..PRrep-:13\}$ 
     $\cup \{PRerr-:1..PRerr-:5\}$ 
     $\rightarrow (\forall ip \in \text{dom}(\text{dests } \xi). ip \in kD(rt \xi) \wedge \text{sqn } (rt \xi) ip \leq \text{the } (\text{dests } \xi ip)))$ "
```

**proof -**

have sqninv:

```
" $\bigwedge \text{dests } rt \text{ rsn } ip.$ 
 $[\forall ip \in \text{dom}(\text{dests}). ip \in kD(rt) \wedge \text{sqn } rt \text{ ip} \leq \text{the } (\text{dests } ip); \text{dests } ip = \text{Some } rsn]$ 
 $\implies \text{sqn } (\text{invalidate } rt \text{ dests}) ip \leq rsn$ "
by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto
```

have indests:

```
" $\bigwedge \text{dests } rt \text{ rsn } ip.$ 
 $[\forall ip \in \text{dom}(\text{dests}). ip \in kD(rt) \wedge \text{sqn } rt \text{ ip} \leq \text{the } (\text{dests } ip); \text{dests } ip = \text{Some } rsn]$ 
 $\implies ip \in kD(rt) \wedge \text{sqn } rt \text{ ip} \leq rsn$ "
by (metis domI option.sel)
```

show ?thesis

```
by inv_cterms
   (clarsimp split: split_if_asm option.split_asm
    elim!: sqninv indests)+
```

qed

Proposition 7.6

**lemma sqns\_increase:**

```
"paadv i  $\models_A$  onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \forall ip. \text{sqn } (rt \xi) ip \leq \text{sqn } (rt \xi') ip)$ "
```

**proof -**

```
{ fix  $\xi :: \text{state}$ 
  assume *: " $\forall ip \in \text{dom}(\text{dests } \xi). ip \in kD (rt \xi) \wedge \text{sqn } (rt \xi) ip \leq \text{the } (\text{dests } \xi ip)$ "
  have " $\forall ip. \text{sqn } (rt \xi) ip \leq \text{sqn } (\text{invalidate } (rt \xi) (\text{dests } \xi)) ip$ "
```

**proof**

```
  fix ip
  from * have " $ip \notin \text{dom}(\text{dests } \xi) \vee \text{sqn } (rt \xi) ip \leq \text{the } (\text{dests } \xi ip)$ " by simp
  thus " $\text{sqn } (rt \xi) ip \leq \text{sqn } (\text{invalidate } (rt \xi) (\text{dests } \xi)) ip$ "
    by (metis domI invalidate_sqn option.sel)
```

qed

```
} note solve_invalidate = this
```

show ?thesis

```
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf addpreRT_welldefined]
    onl_invariant_sterms [OF aadv_wf dests_bigger_than_sqn]
    simp add: solve_invalidate)
```

qed

Proposition 7.7

**lemma ip\_constant:**

```
"paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, \_). ip \xi = i)$ "
by (inv_cterms simp add:  $\sigma_{AODV\_def}$ )
```

Proposition 7.8

**lemma sender\_ip\_valid':**

"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast ( $\lambda m$ . not\_Pkt m  $\longrightarrow$  msg\_sender m = ip  $\xi$ ) a)"  
by inv\_cterms

lemma sender\_ip\_valid:

"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast ( $\lambda m$ . not\_Pkt m  $\longrightarrow$  msg\_sender m = i) a)"  
by (rule step\_invariant\_weaken\_with\_invariantE [OF ip\_constant sender\_ip\_valid'])  
(auto dest!: onlD onllD)

lemma received\_msg\_inv:

"paadv i  $\models$  (recvmsg P  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .  $l \in \{PAadv-:1\} \longrightarrow P$  (msg  $\xi$ ))"  
by inv\_cterms

Proposition 7.9

lemma sip\_not\_ip':

"paadv i  $\models$  (recvmsg ( $\lambda m$ . not\_Pkt m  $\longrightarrow$  msg\_sender m  $\neq$  i)  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_)$ . sip  $\xi \neq$  ip  $\xi$ )"  
by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]  
onl\_invariant\_sterms [OF aadv\_wf ip\_constant [THEN invariant\_restrict\_inD]]  
simp add: clear\_locals\_sip\_not\_ip') clarsimp+

lemma sip\_not\_ip:

"paadv i  $\models$  (recvmsg ( $\lambda m$ . not\_Pkt m  $\longrightarrow$  msg\_sender m  $\neq$  i)  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_)$ . sip  $\xi \neq$  i)"  
by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]  
onl\_invariant\_sterms [OF aadv\_wf ip\_constant [THEN invariant\_restrict\_inD]]  
simp add: clear\_locals\_sip\_not\_ip') clarsimp+

Neither sip\_not\_ip' nor sip\_not\_ip is needed to show loop freedom.

Proposition 7.10

lemma hop\_count\_positive:

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_)$ .  $\forall ip \in kD$  (rt  $\xi$ ). the (dhops (rt  $\xi$ ) ip)  $\geq$  1)"  
by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf addpreRT\_welldefined]) auto

lemma rreq\_dip\_in\_vD\_dip\_eq\_ip:

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ . ( $l \in \{PRreq-:16..PRreq-:18\} \longrightarrow$  dip  $\xi \in vD$ (rt  $\xi$ )  
 $\wedge$  ( $l \in \{PRreq-:5, PRreq-:6\} \longrightarrow$  dip  $\xi =$  ip  $\xi$ )  
 $\wedge$  ( $l \in \{PRreq-:15..PRreq-:18\} \longrightarrow$  dip  $\xi \neq$  ip  $\xi$ ))"

proof (inv\_cterms, elim conjE)

fix l  $\xi$  pp p'

assume "( $\xi, pp$ )  $\in$  reachable (paadv i) TT"

and "{PRreq-:17}[[ $\lambda\xi$ .  $\xi$ (rt := the (addpreRT (rt  $\xi$ ) (oip  $\xi$ ) {the (nhop (rt  $\xi$ ) (dip  $\xi$ ))})]] p'  
 $\in$  sterms  $\Gamma_{AODV}$  pp"

and "l = PRreq-:17"

and "dip  $\xi \in vD$  (rt  $\xi$ )"

from this(1-3) have "oip  $\xi \in kD$  (rt  $\xi$ )"

by (auto dest: onl\_invariant\_sterms [OF aadv\_wf addpreRT\_welldefined, where l="PRreq-:17"])

with (dip  $\xi \in vD$  (rt  $\xi$ ))

show "dip  $\xi \in vD$  (the (addpreRT (rt  $\xi$ ) (oip  $\xi$ ) {the (nhop (rt  $\xi$ ) (dip  $\xi$ ))}))" by simp

qed

lemma rrep\_dip\_in\_vD:

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ . ( $l \in \{PRrep-:4..PRrep-:6\} \longrightarrow$  dip  $\xi \in vD$ (rt  $\xi$ ))"

proof inv\_cterms

fix l  $\xi$  pp p'

assume "( $\xi, pp$ )  $\in$  reachable (paadv i) TT"

and "{PRrep-:5}[[ $\lambda\xi$ .  $\xi$ (rt := the (addpreRT (rt  $\xi$ ) (the (nhop (rt  $\xi$ ) (dip  $\xi$ )) {the (nhop (rt  $\xi$ ) (oip  $\xi$ ))}))]] p'  
 $\in$  sterms  $\Gamma_{AODV}$  pp"

and "l = PRrep-:5"

and "dip  $\xi \in vD$  (rt  $\xi$ )"

from this(1-3) have "the (nhop (rt  $\xi$ ) (dip  $\xi$ ))  $\in kD$  (rt  $\xi$ )"

by (auto dest: onl\_invariant\_sterms [OF aadv\_wf addpreRT\_welldefined, where l="PRrep-:5"])

with (dip  $\xi \in vD$  (rt  $\xi$ ))

show "dip  $\xi \in vD$  (the (addpreRT (rt  $\xi$ ) (the (nhop (rt  $\xi$ ) (dip  $\xi$ )) {the (nhop (rt  $\xi$ ) (oip  $\xi$ ))}))"

by simp

qed

Proposition 7.11

lemma anycast\_msg\_zhops:

```
" $\wedge$ rreqid dip dsn dsk oip osn sip.  
  paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_)$ . anycast msg_zhops a)"  
proof (inv_cterms inv add:  
  onl_invariant_sterms [OF aadv_wf rreq_dip_in_vD_dip_eq_ip]  
  onl_invariant_sterms [OF aadv_wf rrep_dip_in_vD]  
  onl_invariant_sterms [OF aadv_wf hop_count_positive],  
  elim conjE)  
fix l  $\xi$  a pp p' pp'  
assume "( $\xi$ , pp)  $\in$  reachable (paadv i) TT"  
  and "{PRreq-:18}unicast( $\lambda\xi$ . the (nhop (rt  $\xi$ ) (oip  $\xi$ )),  
     $\lambda\xi$ . Rrep (the (dhops (rt  $\xi$ ) (dip  $\xi$ ))) (dip  $\xi$ ) (sqn (rt  $\xi$ ) (dip  $\xi$ )) (oip  $\xi$ ) (ip  $\xi$ )).  
    p'  $\triangleright$  pp'  $\in$  sterms  $\Gamma_{AODV}$  pp)"  
  and "l = PRreq-:18"  
  and "a = unicast (the (nhop (rt  $\xi$ ) (oip  $\xi$ )))  
    (Rrep (the (dhops (rt  $\xi$ ) (dip  $\xi$ ))) (dip  $\xi$ ) (sqn (rt  $\xi$ ) (dip  $\xi$ )) (oip  $\xi$ ) (ip  $\xi$ ))"  
  and *: " $\forall ip \in kD$  (rt  $\xi$ ). Suc 0  $\leq$  the (dhops (rt  $\xi$ ) ip)"  
  and "dip  $\xi \in vD$  (rt  $\xi$ )"  
from (dip  $\xi \in vD$  (rt  $\xi$ )) have "dip  $\xi \in kD$  (rt  $\xi$ )"  
  by (rule vD_id_gives_kD(1))  
with * have "Suc 0  $\leq$  the (dhops (rt  $\xi$ ) (dip  $\xi$ ))" ..  
thus "0 < the (dhops (rt  $\xi$ ) (dip  $\xi$ ))" by simp  
next  
fix l  $\xi$  a pp p' pp'  
assume "( $\xi$ , pp)  $\in$  reachable (paadv i) TT"  
  and "{PRrep-:6}unicast( $\lambda\xi$ . the (nhop (rt  $\xi$ ) (oip  $\xi$ )),  
     $\lambda\xi$ . Rrep (the (dhops (rt  $\xi$ ) (dip  $\xi$ ))) (dip  $\xi$ ) (sqn (rt  $\xi$ ) (dip  $\xi$ )) (oip  $\xi$ ) (ip  $\xi$ )).  
    p'  $\triangleright$  pp'  $\in$  sterms  $\Gamma_{AODV}$  pp)"  
  and "l = PRrep-:6"  
  and "a = unicast (the (nhop (rt  $\xi$ ) (oip  $\xi$ )))  
    (Rrep (the (dhops (rt  $\xi$ ) (dip  $\xi$ ))) (dip  $\xi$ ) (sqn (rt  $\xi$ ) (dip  $\xi$ )) (oip  $\xi$ ) (ip  $\xi$ ))"  
  and *: " $\forall ip \in kD$  (rt  $\xi$ ). Suc 0  $\leq$  the (dhops (rt  $\xi$ ) ip)"  
  and "dip  $\xi \in vD$  (rt  $\xi$ )"  
from (dip  $\xi \in vD$  (rt  $\xi$ )) have "dip  $\xi \in kD$  (rt  $\xi$ )"  
  by (rule vD_id_gives_kD(1))  
with * have "Suc 0  $\leq$  the (dhops (rt  $\xi$ ) (dip  $\xi$ ))" ..  
thus "the (dhops (rt  $\xi$ ) (dip  $\xi$ )) = 0  $\longrightarrow$  dip  $\xi = ip \xi$ "  
  by auto  
qed
```

lemma hop\_count\_zero\_oip\_dip\_sip:

```
"paadv i  $\models$  (recvmmsg msg_zhops  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .  
  (l  $\in$  {PAadv-:4..PAadv-:5}  $\cup$  {PRreq-:n/n. True}  $\rightarrow$   
    (hops  $\xi = 0 \rightarrow$  oip  $\xi = sip \xi$ ))  
   $\wedge$   
  ((l  $\in$  {PAadv-:6..PAadv-:7}  $\cup$  {PRrep-:n/n. True}  $\rightarrow$   
    (hops  $\xi = 0 \rightarrow$  dip  $\xi = sip \xi$ ))))"  
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]) auto
```

lemma osn\_rreq:

```
"paadv i  $\models$  (recvmmsg rreq_rrep_sn  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .  
  l  $\in$  {PAadv-:4, PAadv-:5}  $\cup$  {PRreq-:n/n. True}  $\rightarrow$  1  $\leq$  osn  $\xi$ )"  
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]) clarsimp
```

lemma osn\_rreq':

```
"paadv i  $\models$  (recvmmsg ( $\lambda m$ . rreq_rrep_sn m  $\wedge$  msg_zhops m)  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .  
  l  $\in$  {PAadv-:4, PAadv-:5}  $\cup$  {PRreq-:n/n. True}  $\rightarrow$  1  $\leq$  osn  $\xi$ )"  
proof (rule invariant_weakenE [OF osn_rreq])  
  fix a  
  assume "recvmmsg ( $\lambda m$ . rreq_rrep_sn m  $\wedge$  msg_zhops m) a"  
  thus "recvmmsg rreq_rrep_sn a"
```



by (cases a) simp\_all  
qed

lemma dsn\_rrep:

"paadv i  $\models$  (recvmsg rreq\_rrep\_sn  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ ).  
 $l \in \{PAadv-:6, PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow 1 \leq dsn \xi$ "  
 by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]) clarsimp

lemma dsn\_rrep':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq_rrep\_sn m \wedge msg\_zhops m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ ).  
 $l \in \{PAadv-:6, PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow 1 \leq dsn \xi$ "

proof (rule invariant\_weakenE [OF dsn\_rrep])

fix a  
 assume "recvmsg ( $\lambda m. rreq_rrep\_sn m \wedge msg\_zhops m$ ) a"  
 thus "recvmsg rreq\_rrep\_sn a"  
 by (cases a) simp\_all

qed

lemma hop\_count\_zero\_oip\_dip\_sip':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq_rrep\_sn m \wedge msg\_zhops m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ ).  
 $(l \in \{PAadv-:4..PAadv-:5\} \cup \{PRreq-:n/n. True\} \rightarrow$   
 $(hops \xi = 0 \rightarrow oip \xi = sip \xi))$   
 $\wedge$   
 $((l \in \{PAadv-:6..PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow$   
 $(hops \xi = 0 \rightarrow dip \xi = sip \xi)))$ "

proof (rule invariant\_weakenE [OF hop\_count\_zero\_oip\_dip\_sip])

fix a  
 assume "recvmsg ( $\lambda m. rreq_rrep\_sn m \wedge msg\_zhops m$ ) a"  
 thus "recvmsg msg\_zhops a"  
 by (cases a) simp\_all

qed

Proposition 7.12

lemma zero\_seq\_unk\_hops\_one':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq_rrep\_sn m \wedge msg\_zhops m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_)$ .  
 $\forall dip \in kD(rt \xi). (sqn (rt \xi) dip = 0 \rightarrow sqnf (rt \xi) dip = unk)$   
 $\wedge (sqnf (rt \xi) dip = unk \rightarrow the (dhops (rt \xi) dip) = 1)$   
 $\wedge (the (dhops (rt \xi) dip) = 1 \rightarrow the (nhop (rt \xi) dip) = dip))$ "

proof -

{ fix dip and  $\xi :: state$  and P  
 assume "sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip = 0"  
 and all: " $\forall ip. sqn (rt \xi) ip \leq sqn (invalidate (rt \xi) (dests \xi)) ip$ "  
 and \*: "sqn (rt  $\xi$ ) dip = 0  $\implies$  P  $\xi$  dip"  
 have "P  $\xi$  dip"

proof -

from all have "sqn (rt  $\xi$ ) dip  $\leq$  sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip" ..  
 with (sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip = 0) have "sqn (rt  $\xi$ ) dip = 0" by simp  
 thus "P  $\xi$  dip" by (rule \*)

qed

} note sqn\_invalidate\_zero [elim!] = this

{ fix dsn hops :: nat and sip oip rt and ip dip :: ip

assume " $\forall dip \in kD(rt)$ .  
 $(sqn rt dip = 0 \rightarrow \pi_3(the (rt dip)) = unk) \wedge$   
 $(\pi_3(the (rt dip)) = unk \rightarrow the (dhops rt dip) = Suc 0) \wedge$   
 $(the (dhops rt dip) = Suc 0 \rightarrow the (nhop rt dip) = dip)"$   
 and "hops = 0  $\rightarrow$  sip = dip"  
 and "Suc 0  $\leq$  dsn"  
 and "ip  $\neq$  dip  $\rightarrow ip \in kD(rt)"$   
 hence "the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = Suc 0  $\rightarrow$   
 the (nhop (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = ip"  
 by - (rule update\_cases, auto simp add: sqn\_def dest!: bspec)

} note prreq\_ok1 [simp] = this

```

{ fix ip dsn hops sip oip rt dip
  assume "∀ dip ∈ kD(rt).
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
    and "Suc 0 ≤ dsn"
    and "ip ≠ dip → ip ∈ kD(rt)"
  hence "π3(the (update rt dip (dsn, kno, val, Suc hops, sip, { }) ip)) = unk →
    the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, { }) ip)) = Suc 0"
    by - (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)
} note prreq_ok2 [simp] = this

{ fix ip dsn hops sip oip rt dip
  assume "∀ dip ∈ kD(rt).
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
    and "Suc 0 ≤ dsn"
    and "ip ≠ dip → ip ∈ kD(rt)"
  hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip, { })) ip = 0 →
    π3 (the (update rt dip (dsn, kno, val, Suc hops, sip, { }) ip)) = unk"
    by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
} note prreq_ok3 [simp] = this

{ fix rt sip
  assume "∀ dip ∈ kD rt.
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
  hence "∀ dip ∈ kD rt.
    (sqn (update rt sip (0, unk, val, Suc 0, sip, { })) dip = 0 →
    π3(the (update rt sip (0, unk, val, Suc 0, sip, { }) dip)) = unk)
    ∧ (π3(the (update rt sip (0, unk, val, Suc 0, sip, { }) dip)) = unk →
    the (dhops (update rt sip (0, unk, val, Suc 0, sip, { }) dip)) = Suc 0)
    ∧ (the (dhops (update rt sip (0, unk, val, Suc 0, sip, { }) dip)) = Suc 0 →
    the (nhop (update rt sip (0, unk, val, Suc 0, sip, { }) dip)) = dip)"
    by - (rule update_cases, simp_all add: sqnf_def sqn_def)
} note prreq_ok4 [simp] = this

have prreq_ok5 [simp]: "∧ sip rt.
  π3(the (update rt sip (0, unk, val, Suc 0, sip, { }) sip)) = unk →
  the (dhops (update rt sip (0, unk, val, Suc 0, sip, { }) sip)) = Suc 0"
  by (rule update_cases) simp_all

have prreq_ok6 [simp]: "∧ sip rt.
  sqn (update rt sip (0, unk, val, Suc 0, sip, { })) sip = 0 →
  π3 (the (update rt sip (0, unk, val, Suc 0, sip, { }) sip)) = unk"
  by (rule update_cases) simp_all

show ?thesis
  by (inv_terms inv add: onl_invariant_sterms_TT [OF aadv_wf addpreRT_welldefined]
    onl_invariant_sterms [OF aadv_wf hop_count_zero_oip_dip_sip']
    seq_step_invariant_sterms_TT [OF sqns_increase aadv_wf aadv_trans]
    onl_invariant_sterms [OF aadv_wf osn_rreq']
    onl_invariant_sterms [OF aadv_wf dsn_rrep']) clarsimp+
qed

lemma zero_seq_unk_hops_one:
  "paadv i ⊨ (recvmsg (λ m. rreq_rrep_sn m ∧ msg_zhops m) →) onl ΓAODV (λ (ξ, _).
    ∀ dip ∈ kD(rt ξ). (sqn (rt ξ) dip = 0 → (sqnf (rt ξ) dip = unk
      ∧ the (dhops (rt ξ) dip) = 1
      ∧ the (nhop (rt ξ) dip) = dip)))"
  by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

```

lemma *kD\_unk\_or\_atleast\_one*:

"paadv i  $\models$  (recvm<sub>sg</sub> rreq<sub>rrep</sub><sub>sn</sub>  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ ).  
 $\forall \text{dip} \in \text{kD}(\text{rt } \xi). \pi_3(\text{the } (\text{rt } \xi \text{ dip})) = \text{unk} \vee 1 \leq \pi_2(\text{the } (\text{rt } \xi \text{ dip}))$ "

proof -

```
{ fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
  assume "dsk1 = unk  $\vee$  Suc 0  $\leq$  dsn2"
  hence " $\pi_3(\text{the } (\text{update } \text{rt } \text{sip } (\text{dsn1}, \text{dsk1}, \text{flag1}, \text{hops1}, \text{nhip1}, \text{pre1}) \text{sip})) = \text{unk}$ 
     $\vee$  Suc 0  $\leq$  sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2)) sip"
    unfolding update_def by (cases "dsk1 =unk") (clarsimp split: option.split)+
  } note fromsip [simp] = this
```

```
{ fix dip sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
  assume allkd: " $\forall \text{dip} \in \text{kD}(\text{rt}). \pi_3(\text{the } (\text{rt } \text{dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } \text{rt } \text{dip}$ "
  and **: "dsk1 = unk  $\vee$  Suc 0  $\leq$  dsn2"
  have " $\forall \text{dip} \in \text{kD}(\text{rt}). \pi_3(\text{the } (\text{update } \text{rt } \text{sip } (\text{dsn1}, \text{dsk1}, \text{flag1}, \text{hops1}, \text{nhip1}, \text{pre1}) \text{dip})) = \text{unk}$ 
     $\vee$  Suc 0  $\leq$  sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2)) dip"
    (is " $\forall \text{dip} \in \text{kD}(\text{rt}). ?\text{prop } \text{dip}$ ")
```

proof

```
fix dip
assume "dip  $\in$  kD(rt)"
thus "?prop dip"
proof (cases "dip = sip")
  assume "dip = sip"
  with ** show ?thesis
  by simp
next
  assume "dip  $\neq$  sip"
  with (dip  $\in$  kD(rt)) allkd show ?thesis
  by simp
```

qed

qed

```
} note solve_update [simp] = this
```

```
{ fix dip rt dests
  assume *: " $\forall \text{ip} \in \text{dom}(\text{dests}). \text{ip} \in \text{kD}(\text{rt}) \wedge \text{sqn } \text{rt } \text{ip} \leq \text{the } (\text{dests } \text{ip})$ "
  and **: " $\forall \text{ip} \in \text{kD}(\text{rt}). \pi_3(\text{the } (\text{rt } \text{ip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } \text{rt } \text{ip}$ "
  have " $\forall \text{dip} \in \text{kD}(\text{rt}). \pi_3(\text{the } (\text{rt } \text{dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } (\text{invalidate } \text{rt } \text{dests}) \text{dip}$ "
```

proof

```
fix dip
assume "dip  $\in$  kD(rt)"
with ** have " $\pi_3(\text{the } (\text{rt } \text{dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } \text{rt } \text{dip}$ " ..
thus " $\pi_3(\text{the } (\text{rt } \text{dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } (\text{invalidate } \text{rt } \text{dests}) \text{dip}$ "
```

proof

```
  assume " $\pi_3(\text{the } (\text{rt } \text{dip})) = \text{unk}$ " thus ?thesis ..
```

next

```
  assume "Suc 0  $\leq$  sqn rt dip"
  have "Suc 0  $\leq$  sqn (invalidate rt dests) dip"
  proof (cases "dip  $\in$  dom(dests)")
    assume "dip  $\in$  dom(dests)"
    with * have "sqn rt dip  $\leq$  the (dests dip)" by simp
    with (Suc 0  $\leq$  sqn rt dip) have "Suc 0  $\leq$  the (dests dip)" by simp
    with (dip  $\in$  dom(dests)) (dip  $\in$  kD(rt)) [THEN kD_Some] show ?thesis
    unfolding invalidate_def sqn_def by auto
```

next

```
  assume "dip  $\notin$  dom(dests)"
  with (Suc 0  $\leq$  sqn rt dip) (dip  $\in$  kD(rt)) [THEN kD_Some] show ?thesis
  unfolding invalidate_def sqn_def by auto
```

qed

```
thus ?thesis by (rule disjI2)
```

qed

qed

```
} note solve_invalidate [simp] = this
```

show ?thesis

```

by (inv_cterms inv add: onl_invariant_sterms_TT [OF aadv_wf addpreRT_welldefined]
    onl_invariant_sterms [OF aadv_wf dests_bigger_than_sqn
        [THEN invariant_restrict_inD]]
    onl_invariant_sterms [OF aadv_wf osn_rreq]
    onl_invariant_sterms [OF aadv_wf dsn_rreq]
    simp add: proj3_inv proj2_eq_sqn)

```

qed

Proposition 7.13

lemma rreq\_rrep\_sn\_any\_step\_invariant:

"paadv i  $\models_A$  (recvmsg rreq\_rrep\_sn  $\rightarrow$ ) onll  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_)$ . anycast rreq\_rrep\_sn a)"

proof -

```

have sqnf_kno: "paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .
    ( $l \in \{PRreq-:16..PRreq-:18\} \rightarrow sqnf (rt \xi) (dip \xi) = kno$ ))"
by (inv_cterms inv add: onl_invariant_sterms_TT [OF aadv_wf addpreRT_welldefined])

```

```

have rrep_sqn_greater_dsn: "paadv i  $\models$  (recvmsg rreq_rrep_sn  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .
    ( $l \in \{PRrep-:1 .. PRrep-:6\} \rightarrow 1 \leq sqn (rt \xi) (dip \xi)$ ))"

```

```

by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]
    onl_invariant_sterms_TT [OF aadv_wf addpreRT_welldefined]
    onl_invariant_sterms [OF aadv_wf dsn_rreq])

```

(clarsimp simp: update\_kno\_dsn\_greater\_zero [simplified])

show ?thesis

```

by (inv_cterms inv add: onl_invariant_sterms_TT [OF aadv_wf addpreRT_welldefined]
    onl_invariant_sterms [OF aadv_wf sequence_number_one_or_bigger
        [THEN invariant_restrict_inD]]
    onl_invariant_sterms [OF aadv_wf kD_unk_or_atleast_one]
    onl_invariant_sterms_TT [OF aadv_wf sqnf_kno]
    onl_invariant_sterms [OF aadv_wf osn_rreq]
    onl_invariant_sterms [OF aadv_wf dsn_rreq]
    onl_invariant_sterms [OF aadv_wf rrep_sqn_greater_dsn])

```

(auto simp: proj2\_eq\_sqn)

qed

Proposition 7.14

lemma rreq\_rrep\_fresh\_any\_step\_invariant:

"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast (rreq\_rrep\_fresh (rt  $\xi$ )) a)"

proof -

```

have rreq_oip: "paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .
    ( $l \in \{PRreq-:3, PRreq-:4, PRreq-:15, PRreq-:27\}$ 
     $\rightarrow oip \xi \in kD(rt \xi)$ 
     $\wedge (sqn (rt \xi) (oip \xi) > (osn \xi)$ 
     $\vee (sqn (rt \xi) (oip \xi) = (osn \xi)$ 
     $\wedge the (dhops (rt \xi) (oip \xi)) \leq Suc (hops \xi)$ 
     $\wedge the (flag (rt \xi) (oip \xi)) = val))))"$ 

```

proof inv\_cterms

fix l  $\xi$  l' pp p'

assume "( $\xi, pp$ )  $\in$  reachable (paadv i) TT"

```

and "{PRreq-:2}  $\ll$   $\lambda\xi. \xi$  (|rt :=
    update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , {})|)  $\ll$  p'  $\in$  sterms  $\Gamma_{AODV}$  pp"
and "l' = PRreq-:3"

```

```

show "osn  $\xi$  < sqn (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , {})) (oip  $\xi$ )
 $\vee$  (sqn (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , {})) (oip  $\xi$ ) = osn  $\xi$ 
 $\wedge$  the (dhops (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , {})) (oip  $\xi$ )
     $\leq$  Suc (hops  $\xi$ )
 $\wedge$  the (flag (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , {})) (oip  $\xi$ )
    = val)"

```

```

unfolding update_def by (clarsimp split: option.split)
(metis linorder_neqE_nat not_less)

```

qed

```

have rrep_prrep: "paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .
    ( $l \in \{PRrep-:4..PRrep-:6\} \rightarrow (dip \xi \in kD(rt \xi)$ 
     $\wedge the (flag (rt \xi) (dip \xi)) = val))))"$ 

```

```

by (inv_terms inv add: onl_invariant_sterms_TT [OF aadv_wf addpreRT_welldefined]
    onl_invariant_sterms [OF aadv_wf sip_in_kD])
show ?thesis
by (inv_terms inv add: onl_invariant_sterms [OF aadv_wf rreq_oip]
    onl_invariant_sterms [OF aadv_wf rreq_dip_in_vD_dip_eq_ip]
    onl_invariant_sterms [OF aadv_wf rrep_prrep])
qed

```

Proposition 7.15

```

lemma rerr_invalid_any_step_invariant:
  "paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast (rerr_invalid (rt  $\xi$ )) a)"
proof -
  have dests_inv: "paadv i  $\models$ 
    onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ . ( $l \in \{PAadv-:15, PPkt-:7, PRreq-:9,$ 
      PRreq-:21, PRrep-:9, PRerr-:1}
       $\longrightarrow (\forall ip \in \text{dom}(\text{dests } \xi). ip \in vD(\text{rt } \xi))$ )
     $\wedge (l \in \{PAadv-:16..PAadv-:19\}$ 
       $\cup \{PPkt-:8..PPkt-:11\}$ 
       $\cup \{PRreq-:10..PRreq-:13\}$ 
       $\cup \{PRreq-:22..PRreq-:25\}$ 
       $\cup \{PRrep-:10..PRrep-:13\}$ 
       $\cup \{PRerr-:2..PRerr-:5\} \longrightarrow (\forall ip \in \text{dom}(\text{dests } \xi). ip \in iD(\text{rt } \xi)$ 
         $\wedge \text{the}(\text{dests } \xi \text{ } ip) = \text{sqn}(\text{rt } \xi) \text{ } ip))$ )
     $\wedge (l = PPkt-:14 \longrightarrow dip \xi \in iD(\text{rt } \xi))$ )"
  by inv_terms (clarsimp split: split_if_asm option.split_asm simp: domIff)+
show ?thesis
by (inv_terms inv add: onl_invariant_sterms [OF aadv_wf dests_inv])
qed

```

Proposition 7.16

Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs  $(rip, rsn)$  is assigned to the variable  $dests$  of type  $ip \rightarrow sqn$ , or to the first argument of the function  $rerr$ , this set is a partial function, i.e., there is at most one entry  $(rip, rsn)$  for each destination  $rip$ : guaranteed by type.

```

lemma dests_vD_inc_sqn:
  "paadv i  $\models$ 
    onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ . ( $l \in \{PAadv-:15, PPkt-:7, PRreq-:9, PRreq-:21, PRrep-:9\}$ 
       $\longrightarrow (\forall ip \in \text{dom}(\text{dests } \xi). ip \in vD(\text{rt } \xi) \wedge \text{the}(\text{dests } \xi \text{ } ip) = \text{inc}(\text{sqn}(\text{rt } \xi) \text{ } ip))$ )
     $\wedge (l = PRerr-:1$ 
       $\longrightarrow (\forall ip \in \text{dom}(\text{dests } \xi). ip \in vD(\text{rt } \xi) \wedge \text{the}(\text{dests } \xi \text{ } ip) > \text{sqn}(\text{rt } \xi) \text{ } ip))$ )"
  by inv_terms (clarsimp split: split_if_asm option.split_asm)+

```

Proposition 7.27

```

lemma route_tables_fresher:
  "paadv i  $\models_A$  (recvmgs rreq_rrep_sn  $\rightarrow$ ) onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_))$ .
     $\forall dip \in kD(\text{rt } \xi). \text{rt } \xi \sqsubseteq_{dip} \text{rt } \xi')$ ")"
proof (inv_terms inv add:
  onl_invariant_sterms [OF aadv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aadv_wf hop_count_positive [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aadv_wf osn_rreq]
  onl_invariant_sterms [OF aadv_wf dsn_rrep]
  onl_invariant_sterms [OF aadv_wf addpreRT_welldefined [THEN invariant_restrict_inD]])
  fix  $\xi$  pp p'
  assume " $(\xi, pp) \in \text{reachable}(\text{paadv } i) (\text{recvmgs } rreq\_rrep\_sn)$ "
  and " $\{PRreq-:2\}[\lambda\xi. \xi(\text{rt} := \text{update}(\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops } \xi), \text{sip } \xi, \{\}))]$ "

```

```

      p' ∈ sterms ΓAODV pp"
    and "Suc 0 ≤ osn ξ"
    and *: "∀ip∈kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) ip)"
  show "∀ip∈kD (rt ξ). rt ξ ⊆ip update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})"
  proof
    fix ip
    assume "ip∈kD (rt ξ)"
    moreover with * have "1 ≤ the (dhops (rt ξ) ip)" by simp
    moreover from ⟨Suc 0 ≤ osn ξ⟩
      have "update_arg_wf (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})" ..
    ultimately show "rt ξ ⊆ip update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})"
      by (rule rt_fresher_update)
  qed
next
fix ξ pp p'
assume "(ξ, pp) ∈ reachable (paadv i) (recvmmsg rreq_rrep_sn)"
  and "{PRrep:0}[[λξ. ξ(rt := update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {}))]]
  p' ∈ sterms ΓAODV pp"
  and "Suc 0 ≤ dsn ξ"
  and *: "∀ip∈kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) ip)"
show "∀ip∈kD (rt ξ). rt ξ ⊆ip update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {})"
proof
  fix ip
  assume "ip∈kD (rt ξ)"
  moreover with * have "1 ≤ the (dhops (rt ξ) ip)" by simp
  moreover from ⟨Suc 0 ≤ dsn ξ⟩
    have "update_arg_wf (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {})" ..
  ultimately show "rt ξ ⊆ip update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {})"
    by (rule rt_fresher_update)
  qed
qed
end

```

## 2.7 The quality increases predicate

```

theory B_Quality_Increases
imports B_Aodv_Predicates B_Fresher
begin

```

```

definition quality_increases :: "state ⇒ state ⇒ bool"
where "quality_increases ξ ξ' ≡ (∀dip∈kD(rt ξ). dip ∈ kD(rt ξ') ∧ rt ξ ⊆dip rt ξ')
      ∧ (∀dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip)"

```

```

lemma quality_increasesI [intro!]:
  assumes "∧dip. dip ∈ kD(rt ξ) ⇒ dip ∈ kD(rt ξ')"
    and "∧dip. [ dip ∈ kD(rt ξ); dip ∈ kD(rt ξ') ] ⇒ rt ξ ⊆dip rt ξ'"
    and "∧dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip"
  shows "quality_increases ξ ξ'"
  unfolding quality_increases_def using assms by clarsimp

```

```

lemma quality_increasesE [elim]:
  fixes dip
  assumes "quality_increases ξ ξ'"
    and "dip∈kD(rt ξ)"
    and "[ dip ∈ kD(rt ξ'); rt ξ ⊆dip rt ξ'; sqn (rt ξ) dip ≤ sqn (rt ξ') dip ] ⇒ R dip ξ ξ'"
  shows "R dip ξ ξ'"
  using assms unfolding quality_increases_def by clarsimp

```

```

lemma quality_increases_rt_fresherD [dest]:
  fixes ip
  assumes "quality_increases ξ ξ'"
    and "ip∈kD(rt ξ)"
  shows "rt ξ ⊆ip rt ξ'"

```

```

using assms by auto

lemma quality_increases_sqnE [elim]:
  fixes dip
  assumes "quality_increases  $\xi \xi'$ "
    and "sqn (rt  $\xi$ ) dip  $\leq$  sqn (rt  $\xi'$ ) dip  $\implies$  R dip  $\xi \xi'$ "
  shows "R dip  $\xi \xi'$ "
  using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_refl [intro, simp]: "quality_increases  $\xi \xi$ "
  by rule simp_all

lemma strictly_fresher_quality_increases_right [elim]:
  fixes  $\sigma \sigma'$  dip
  assumes "rt ( $\sigma$  i)  $\sqsubseteq_{\text{dip}}$  rt ( $\sigma$  nhip)"
    and qinc: "quality_increases ( $\sigma$  nhip) ( $\sigma'$  nhip)"
    and "dip  $\in$  kD(rt ( $\sigma$  nhip))"
  shows "rt ( $\sigma$  i)  $\sqsubseteq_{\text{dip}}$  rt ( $\sigma'$  nhip)"
proof -
  from qinc have "rt ( $\sigma$  nhip)  $\sqsubseteq_{\text{dip}}$  rt ( $\sigma'$  nhip)" using (dip  $\in$  kD(rt ( $\sigma$  nhip)))
  by auto
  with (rt ( $\sigma$  i)  $\sqsubseteq_{\text{dip}}$  rt ( $\sigma$  nhip)) show ?thesis ..
qed

lemma kD_quality_increases [elim]:
  assumes "i  $\in$  kD(rt  $\xi$ )"
    and "quality_increases  $\xi \xi'$ "
  shows "i  $\in$  kD(rt  $\xi')$ "
  using assms by auto

lemma kD_nsqn_quality_increases [elim]:
  assumes "i  $\in$  kD(rt  $\xi$ )"
    and "quality_increases  $\xi \xi'$ "
  shows "i  $\in$  kD(rt  $\xi')$   $\wedge$  nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi')$  i"
proof -
  from assms have "i  $\in$  kD(rt  $\xi')$ " ..
  moreover with assms have "rt  $\xi \sqsubseteq_i$  rt  $\xi'$ " by auto
  ultimately have "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi')$  i"
  using (i  $\in$  kD(rt  $\xi$ )) by - (erule(2) rt_fresher_imp_nsqn_le)
  with (i  $\in$  kD(rt  $\xi')$ ) show ?thesis ..
qed

lemma nsqn_quality_increases [elim]:
  assumes "i  $\in$  kD(rt  $\xi$ )"
    and "quality_increases  $\xi \xi'$ "
  shows "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi')$  i"
  using assms by (rule kD_nsqn_quality_increases [THEN conjunct2])

lemma kD_nsqn_quality_increases_trans [elim]:
  assumes "i  $\in$  kD(rt  $\xi$ )"
    and "s  $\leq$  nsqn (rt  $\xi$ ) i"
    and "quality_increases  $\xi \xi'$ "
  shows "i  $\in$  kD(rt  $\xi')$   $\wedge$  s  $\leq$  nsqn (rt  $\xi')$  i"
proof
  from (i  $\in$  kD(rt  $\xi$ )) and (quality_increases  $\xi \xi'$ ) show "i  $\in$  kD(rt  $\xi')$ " ..
next
  from (i  $\in$  kD(rt  $\xi$ )) and (quality_increases  $\xi \xi'$ ) have "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi')$  i" ..
  with (s  $\leq$  nsqn (rt  $\xi$ ) i) show "s  $\leq$  nsqn (rt  $\xi')$  i" by (rule le_trans)
qed

lemma nsqn_quality_increases_nsqn_lt_lt [elim]:
  assumes "i  $\in$  kD(rt  $\xi$ )"
    and "quality_increases  $\xi \xi'$ "
    and "s < nsqn (rt  $\xi$ ) i"

```

```

shows "s < nsqn (rt ξ') i"
proof -
  from assms(1-2) have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
  with ⟨s < nsqn (rt ξ) i⟩ show "s < nsqn (rt ξ') i" by simp
qed

lemma nsqn_quality_increases_dhops [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "nsqn (rt ξ) i = nsqn (rt ξ') i"
  shows "the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i)"
  using assms unfolding quality_increases_def
  by (clarsimp) (drule(1) bspec, clarsimp simp: rt_fresher_def2)

lemma nsqn_quality_increases_nsqn_eq_le [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "s = nsqn (rt ξ) i"
  shows "s < nsqn (rt ξ') i ∨ (s = nsqn (rt ξ') i ∧ the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i))"
  using assms by (metis nat_less_le nsqn_quality_increases nsqn_quality_increases_dhops)

lemma quality_increases_rreq_rrep_props [elim]:
  fixes sn ip hops sip
  assumes qinc: "quality_increases (σ sip) (σ' sip)"
    and "1 ≤ sn"
    and *: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
      ∧ (nsqn (rt (σ sip)) ip = sn
        → (the (dhops (rt (σ sip)) ip) ≤ hops
          ∨ the (flag (rt (σ sip)) ip) = inv))"
  shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip
    ∧ (nsqn (rt (σ' sip)) ip = sn
      → (the (dhops (rt (σ' sip)) ip) ≤ hops
        ∨ the (flag (rt (σ' sip)) ip) = inv))"
  (is "_ ∧ ?nsqnafter")
proof -
  from * obtain "ip ∈ kD(rt (σ sip))" and "sn ≤ nsqn (rt (σ sip)) ip" by auto

  from ⟨quality_increases (σ sip) (σ' sip)⟩
  have "sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip" ..
  from ⟨quality_increases (σ sip) (σ' sip)⟩ and ⟨ip ∈ kD (rt (σ sip))⟩
  have "ip ∈ kD (rt (σ' sip))" ..

  from ⟨sn ≤ nsqn (rt (σ sip)) ip⟩ have ?nsqnafter
proof
  assume "sn < nsqn (rt (σ sip)) ip"
  also from ⟨ip ∈ kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  have "... ≤ nsqn (rt (σ' sip)) ip" ..
  finally have "sn < nsqn (rt (σ' sip)) ip" .
  thus ?thesis by simp
next
  assume "sn = nsqn (rt (σ sip)) ip"
  with ⟨ip ∈ kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  have "sn < nsqn (rt (σ' sip)) ip
    ∨ (sn = nsqn (rt (σ' sip)) ip
      ∧ the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip))" ..
  hence "sn < nsqn (rt (σ' sip)) ip
    ∨ (nsqn (rt (σ' sip)) ip = sn ∧ (the (dhops (rt (σ' sip)) ip) ≤ hops
      ∨ the (flag (rt (σ' sip)) ip) = inv))"
proof
  assume "sn < nsqn (rt (σ' sip)) ip" thus ?thesis ..
next
  assume "sn = nsqn (rt (σ' sip)) ip
    ∧ the (dhops (rt (σ sip)) ip) ≥ the (dhops (rt (σ' sip)) ip)"
  hence "sn = nsqn (rt (σ' sip)) ip"

```



```

and "the (dhops (rt ( $\sigma'$  sip)) ip)  $\leq$  the (dhops (rt ( $\sigma$  sip)) ip)" by auto

from * and  $\langle sn = nsqn (rt (\sigma sip)) ip \rangle$  have "the (dhops (rt ( $\sigma sip$ )) ip)  $\leq$  hops
                                          $\vee$  the (flag (rt ( $\sigma sip$ )) ip) = inv"

by simp
thus ?thesis
proof
  assume "the (dhops (rt ( $\sigma sip$ )) ip)  $\leq$  hops"
  with  $\langle$ the (dhops (rt ( $\sigma' sip$ )) ip)  $\leq$  the (dhops (rt ( $\sigma sip$ )) ip) $\rangle$ 
  have "the (dhops (rt ( $\sigma' sip$ )) ip)  $\leq$  hops" by simp
  with  $\langle sn = nsqn (rt (\sigma' sip)) ip \rangle$  show ?thesis by simp
next
  assume "the (flag (rt ( $\sigma sip$ )) ip) = inv"
  with  $\langle ip \in kD(rt (\sigma sip)) \rangle$  have "nsqn (rt ( $\sigma sip$ )) ip = sqn (rt ( $\sigma sip$ )) ip - 1" ..

with  $\langle sn \geq 1 \rangle$  and  $\langle sn = nsqn (rt (\sigma sip)) ip \rangle$ 
  have "sqn (rt ( $\sigma sip$ )) ip  $>$  1" by simp

from  $\langle ip \in kD(rt (\sigma' sip)) \rangle$  show ?thesis
proof (rule vD_or_iD)
  assume "ip  $\in$  iD(rt ( $\sigma' sip$ ))"
  hence "the (flag (rt ( $\sigma' sip$ )) ip) = inv" ..
  with  $\langle sn = nsqn (rt (\sigma' sip)) ip \rangle$  show ?thesis
  by simp
next
  assume "ip  $\in$  vD(rt ( $\sigma' sip$ ))"
  hence "nsqn (rt ( $\sigma' sip$ )) ip = sqn (rt ( $\sigma' sip$ )) ip" ..
  with  $\langle sqn (rt (\sigma sip)) ip \leq sqn (rt (\sigma' sip)) ip \rangle$ 
  have "nsqn (rt ( $\sigma' sip$ )) ip  $\geq$  sqn (rt ( $\sigma sip$ )) ip" by simp

with  $\langle sqn (rt (\sigma sip)) ip > 1 \rangle$ 
  have "nsqn (rt ( $\sigma' sip$ )) ip  $>$  sqn (rt ( $\sigma sip$ )) ip - 1" by simp
with  $\langle nsqn (rt (\sigma sip)) ip = sqn (rt (\sigma sip)) ip - 1 \rangle$ 
  have "nsqn (rt ( $\sigma' sip$ )) ip  $>$  nsqn (rt ( $\sigma sip$ )) ip" by simp
with  $\langle sn = nsqn (rt (\sigma sip)) ip \rangle$  have "nsqn (rt ( $\sigma' sip$ )) ip  $>$  sn"
  by simp
  thus ?thesis ..
qed
qed
qed
thus ?thesis by (metis (mono_tags) le_cases not_le)
qed
with  $\langle ip \in kD (rt (\sigma' sip)) \rangle$  show "ip  $\in$  kD (rt ( $\sigma' sip$ ))  $\wedge$  ?nsqnafter" ..
qed

```

lemma quality\_increases\_rreq\_rrep\_props':

```

fixes sn ip hops sip
assumes " $\forall j$ . quality_increases ( $\sigma j$ ) ( $\sigma' j$ )"
and "1  $\leq$  sn"
and *: "ip  $\in$  kD(rt ( $\sigma sip$ ))  $\wedge$  sn  $\leq$  nsqn (rt ( $\sigma sip$ )) ip
         $\wedge$  (nsqn (rt ( $\sigma sip$ )) ip = sn
             $\rightarrow$  (the (dhops (rt ( $\sigma sip$ )) ip)  $\leq$  hops
                 $\vee$  the (flag (rt ( $\sigma sip$ )) ip) = inv))"
shows "ip  $\in$  kD(rt ( $\sigma' sip$ ))  $\wedge$  sn  $\leq$  nsqn (rt ( $\sigma' sip$ )) ip
         $\wedge$  (nsqn (rt ( $\sigma' sip$ )) ip = sn
             $\rightarrow$  (the (dhops (rt ( $\sigma' sip$ )) ip)  $\leq$  hops
                 $\vee$  the (flag (rt ( $\sigma' sip$ )) ip) = inv))"

```

proof -

```

from assms(1) have "quality_increases ( $\sigma sip$ ) ( $\sigma' sip$ )" ..
thus ?thesis using assms(2-3) by (rule quality_increases_rreq_rrep_props)
qed

```

lemma rteq\_quality\_increases:

```

assumes "∀j. j ≠ i → quality_increases (σ j) (σ' j)"
and "rt (σ' i) = rt (σ i)"
shows "∀j. quality_increases (σ j) (σ' j)"
using assms by clarsimp (metis order_refl quality_increasesI rt_fresher_refl)

```

```

definition msg_fresh :: "(ip ⇒ state) ⇒ msg ⇒ bool"

```

```

where "msg_fresh σ m ≡

```

```

  case m of Rreq hopsc _ _ _ oipc osnc sipc ⇒ osnc ≥ 1 ∧ (sipc ≠ oipc →
    oipc ∈ kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) oipc ≥ osnc
    ∧ (nsqn (rt (σ sipc)) oipc = osnc
      → (hopsc ≥ the (dhops (rt (σ sipc)) oipc)
        ∨ the (flag (rt (σ sipc)) oipc) = inv)))
  | Rrep hopsc dipc dsnc _ sipc ⇒ dsnc ≥ 1 ∧ (sipc ≠ dipc →
    dipc ∈ kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) dipc ≥ dsnc
    ∧ (nsqn (rt (σ sipc)) dipc = dsnc
      → (hopsc ≥ the (dhops (rt (σ sipc)) dipc)
        ∨ the (flag (rt (σ sipc)) dipc) = inv)))
  | Rerr destsc sipc ⇒ (∀ripc ∈ dom(destsc). (ripc ∈ kD(rt (σ sipc))
    ∧ the (destsc ripc) - 1 ≤ nsqn (rt (σ sipc)) ripc))
  | _ ⇒ True"

```

```

lemma msg_fresh [simp]:

```

```

"∧hops rreqid dip dsn dsk oip osn sip.
  msg_fresh σ (Rreq hops rreqid dip dsn dsk oip osn sip) =
    (osn ≥ 1 ∧ (sip ≠ oip → oip ∈ kD(rt (σ sip))
      ∧ nsqn (rt (σ sip)) oip ≥ osn
      ∧ (nsqn (rt (σ sip)) oip = osn
        → (hops ≥ the (dhops (rt (σ sip)) oip)
          ∨ the (flag (rt (σ sip)) oip) = inv))))))"
"∧hops dip dsn oip sip. msg_fresh σ (Rrep hops dip dsn oip sip) =
  (dsn ≥ 1 ∧ (sip ≠ dip → dip ∈ kD(rt (σ sip))
    ∧ nsqn (rt (σ sip)) dip ≥ dsn
    ∧ (nsqn (rt (σ sip)) dip = dsn
      → (hops ≥ the (dhops (rt (σ sip)) dip)
        ∨ the (flag (rt (σ sip)) dip) = inv)))))"
"∧dests sip. msg_fresh σ (Rerr dests sip) =
  (∀ripc ∈ dom(dests). (ripc ∈ kD(rt (σ sip))
    ∧ the (dests ripc) - 1 ≤ nsqn (rt (σ sip)) ripc))"
"∧d dip. msg_fresh σ (Newpkt d dip) = True"
"∧dip sip. msg_fresh σ (Pkt d dip sip) = True"
unfolding msg_fresh_def by simp_all

```

```

lemma msg_fresh_inc_sn [simp, elim]:

```

```

"msg_fresh σ m ⇒ rreq_rrep_sn m"
by (cases m) simp_all

```

```

lemma recv_msg_fresh_inc_sn [simp, elim]:

```

```

"orecvmsg (msg_fresh) σ m ⇒ recvmmsg rreq_rrep_sn m"
by (cases m) simp_all

```

```

lemma rreq_nsqn_is_fresh [simp]:

```

```

fixes σ msg hops rreqid dip dsn dsk oip osn sip
assumes "rreq_rrep_fresh (rt (σ sip)) (Rreq hops rreqid dip dsn dsk oip osn sip)"
and "rreq_rrep_sn (Rreq hops rreqid dip dsn dsk oip osn sip)"
shows "msg_fresh σ (Rreq hops rreqid dip dsn dsk oip osn sip)"
(is "msg_fresh σ ?msg")

```

```

proof -

```

```

  let ?rt = "rt (σ sip)"
  from assms(2) have "1 ≤ osn" by simp
  thus ?thesis
  unfolding msg_fresh_def
  proof (simp only: msg.case, intro conjI impI)
    assume "sip ≠ oip"
    with assms(1) show "oip ∈ kD(?rt)" by simp
  qed

```

```

next
  assume "sip ≠ oip"
    and "nsqn ?rt oip = osn"
  show "the (dhops ?rt oip) ≤ hops ∨ the (flag ?rt oip) = inv"
  proof (cases "oip ∈ vD(?rt)")
    assume "oip ∈ vD(?rt)"
    hence "nsqn ?rt oip = sqn ?rt oip" ..
    with ⟨nsqn ?rt oip = osn⟩ have "sqn ?rt oip = osn" by simp
    with assms(1) and ⟨sip ≠ oip⟩ have "the (dhops ?rt oip) ≤ hops"
      by simp
    thus ?thesis ..
  next
    assume "oip ∉ vD(?rt)"
    moreover from assms(1) and ⟨sip ≠ oip⟩ have "oip ∈ kD(?rt)" by simp
    ultimately have "oip ∈ iD(?rt)" by auto
    hence "the (flag ?rt oip) = inv" ..
    thus ?thesis ..
  qed
next
  assume "sip ≠ oip"
  with assms(1) have "osn ≤ sqn ?rt oip" by auto
  thus "osn ≤ nsqn (rt (σ sip)) oip"
  proof (rule nat_le_eq_or_lt)
    assume "osn < sqn ?rt oip"
    hence "osn ≤ sqn ?rt oip - 1" by simp
    also have "... ≤ nsqn ?rt oip" by (rule sqn_nsqn)
    finally show "osn ≤ nsqn ?rt oip" .
  next
    assume "osn = sqn ?rt oip"
    with assms(1) and ⟨sip ≠ oip⟩ have "oip ∈ kD(?rt)"
      and "the (flag ?rt oip) = val"
    by auto
    hence "nsqn ?rt oip = sqn ?rt oip" ..
    with ⟨osn = sqn ?rt oip⟩ have "nsqn ?rt oip = osn" by simp
    thus "osn ≤ nsqn ?rt oip" by simp
  qed
qed simp
qed

lemma rrep_nsqn_is_fresh [simp]:
  fixes σ msg hops dip dsn oip sip
  assumes "rreq_rrep_fresh (rt (σ sip)) (Rrep hops dip dsn oip sip)"
    and "rreq_rrep_sn (Rrep hops dip dsn oip sip)"
  shows "msg_fresh σ (Rrep hops dip dsn oip sip)"
    (is "msg_fresh σ ?msg")
  proof -
    let ?rt = "rt (σ sip)"
    from assms have "sip ≠ dip → dip ∈ kD(?rt) ∧ sqn ?rt dip = dsn ∧ the (flag ?rt dip) = val"
      by simp
    hence "sip ≠ dip → dip ∈ kD(?rt) ∧ nsqn ?rt dip ≥ dsn"
      by clarsimp
    with assms show "msg_fresh σ ?msg"
      by clarsimp
  qed

lemma rerr_nsqn_is_fresh [simp]:
  fixes σ msg dests sip
  assumes "rerr_invalid (rt (σ sip)) (Rerr dests sip)"
  shows "msg_fresh σ (Rerr dests sip)"
    (is "msg_fresh σ ?msg")
  proof -
    let ?rt = "rt (σ sip)"
    from assms have *: "(∀ rip ∈ dom(dests). (rip ∈ iD(rt (σ sip))
      ∧ the (dests rip) = sqn (rt (σ sip)) rip))"

```

```

  by clarsimp
  have "( $\forall rip \in \text{dom}(\text{dests}). (\text{rip} \in \text{kD}(\text{rt}(\sigma \text{ sip})) \wedge \text{the}(\text{dests } rip) - 1 \leq \text{nsqn}(\text{rt}(\sigma \text{ sip})) \text{ rip}))"$ )"
proof
  fix rip
  assume "rip  $\in$  dom dests"
  with * have "rip  $\in$  iD(rt ( $\sigma$  sip))" and "the (dests rip) = sqn (rt ( $\sigma$  sip)) rip"
  by auto

  from this(2) have "the (dests rip) - 1 = sqn (rt ( $\sigma$  sip)) rip - 1" by simp
  also have "...  $\leq$  nsqn (rt ( $\sigma$  sip)) rip" by (rule sqn_nsqn)
  finally have "the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip" .

  with (rip  $\in$  iD(rt ( $\sigma$  sip)))
  show "rip  $\in$  kD(rt ( $\sigma$  sip))  $\wedge$  the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip"
  by clarsimp
qed
thus "msg_fresh  $\sigma$  ?msg"
  by simp
qed

lemma quality_increases_msg_fresh [elim]:
  assumes qinc: " $\forall j. \text{quality\_increases}(\sigma j)(\sigma' j)"$ "
  and "msg_fresh  $\sigma m$ "
  shows "msg_fresh  $\sigma' m$ "
using assms(2)
proof (cases m)
  fix hops rreqid dip dsn dsk oip osn sip
  assume [simp]: "m = Rreq hops rreqid dip dsn dsk oip osn sip"
  and "msg_fresh  $\sigma m$ "
  then have "osn  $\geq$  1" and "sip = oip  $\vee$  (oip  $\in$  kD(rt ( $\sigma$  sip))  $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma$  sip)) oip
     $\wedge$  (nsqn (rt ( $\sigma$  sip)) oip = osn
     $\longrightarrow$  (the (dhops (rt ( $\sigma$  sip)) oip)  $\leq$  hops
     $\vee$  the (flag (rt ( $\sigma$  sip)) oip) = inv)))"

  by auto
  from this(2) show ?thesis
proof
  assume "sip = oip" with (osn  $\geq$  1) show ?thesis by simp
next
  assume "oip  $\in$  kD(rt ( $\sigma$  sip))  $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma$  sip)) oip
     $\wedge$  (nsqn (rt ( $\sigma$  sip)) oip = osn
     $\longrightarrow$  (the (dhops (rt ( $\sigma$  sip)) oip)  $\leq$  hops
     $\vee$  the (flag (rt ( $\sigma$  sip)) oip) = inv))"
  moreover from qinc have "quality_increases ( $\sigma$  sip) ( $\sigma'$  sip)" ..
  ultimately have "oip  $\in$  kD(rt ( $\sigma'$  sip))  $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma'$  sip)) oip
     $\wedge$  (nsqn (rt ( $\sigma'$  sip)) oip = osn
     $\longrightarrow$  (the (dhops (rt ( $\sigma'$  sip)) oip)  $\leq$  hops
     $\vee$  the (flag (rt ( $\sigma'$  sip)) oip) = inv))"

  using (osn  $\geq$  1) by (rule quality_increases_rreq_rrep_props [rotated 2])
  with (osn  $\geq$  1) show "msg_fresh  $\sigma' m$ "
  by (clarsimp)
qed
next
  fix hops dip dsn oip sip
  assume [simp]: "m = Rrep hops dip dsn oip sip"
  and "msg_fresh  $\sigma m$ "
  then have "dsn  $\geq$  1" and "sip = dip  $\vee$  (dip  $\in$  kD(rt ( $\sigma$  sip))  $\wedge$  dsn  $\leq$  nsqn (rt ( $\sigma$  sip)) dip
     $\wedge$  (nsqn (rt ( $\sigma$  sip)) dip = dsn
     $\longrightarrow$  (the (dhops (rt ( $\sigma$  sip)) dip)  $\leq$  hops
     $\vee$  the (flag (rt ( $\sigma$  sip)) dip) = inv)))"

  by auto
  from this(2) show "?thesis"
proof
  assume "sip = dip" with (dsn  $\geq$  1) show ?thesis by simp

```

```

next
  assume "dip∈kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip
        ∧ (nsqn (rt (σ sip)) dip = dsn
           → (the (dhops (rt (σ sip)) dip) ≤ hops
              ∨ the (flag (rt (σ sip)) dip) = inv))"
  moreover from qinc have "quality_increases (σ sip) (σ' sip)" ..
  ultimately have "dip∈kD(rt (σ' sip)) ∧ dsn ≤ nsqn (rt (σ' sip)) dip
                  ∧ (nsqn (rt (σ' sip)) dip = dsn
                     → (the (dhops (rt (σ' sip)) dip) ≤ hops
                        ∨ the (flag (rt (σ' sip)) dip) = inv))"

    using (dsn ≥ 1) by (rule quality_increases_rreq_rrep_props [rotated 2])
  with (dsn ≥ 1) show "msg_fresh σ' m"
    by clarsimp
qed
next
fix dests sip
assume [simp]: "m = Rerr dests sip"
  and "msg_fresh σ m"
then have *: "∀rip∈dom(dests). rip∈kD(rt (σ sip))
             ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"

  by simp
have "∀rip∈dom(dests). rip∈kD(rt (σ' sip))
     ∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip"

proof
fix rip
assume "rip∈dom(dests)"
with * have "rip∈kD(rt (σ sip))" and "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
  by - (drule(1) bspec, clarsimp)+
moreover from qinc have "quality_increases (σ sip) (σ' sip)" by simp
ultimately show "rip∈kD(rt (σ' sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip" ..
qed
thus ?thesis by simp
qed simp_all
end

```

## 2.8 The ‘open’ AODV model

```

theory B_OAodv
imports B_Aodv "../.../AWN/OAWN_SOS_Labels" "../.../AWN/OAWN_Convert"
begin

```

Definitions for stating and proving global network properties over individual processes.

```

definition σAODV' :: "(ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp) set"
where "σAODV' ≡ {(λi. aodv_init i, ΓAODV PAodv)}"

```

abbreviation opaodv

```

:: "ip ⇒ ((ip ⇒ state) × (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
where
"opaodv i ≡ (| init = σAODV', trans = oseqp_sos ΓAODV i |)"

```

```

lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))"
  unfolding σAODV'_def σAODV'_def by rule simp_all

```

```

lemma oaodv_control_within [simp]: "control_within ΓAODV (init (opaodv i))"
  unfolding σAODV'_def by (rule control_withinI) (auto simp del: ΓAODV_simps)

```

```

lemma σAODV'_labels [simp]: "(σ, p) ∈ σAODV' ⇒ labels ΓAODV p = {PAodv-.0}"
  unfolding σAODV'_def by simp

```

```

lemma oaodv_init_kD_empty [simp]:
  "(σ, p) ∈ σAODV' ⇒ kD (rt (σ i)) = {}"
  unfolding σAODV'_def kD_def by simp

```

```

lemma oaodv_init_vD_empty [simp]:
  "( $\sigma, p$ )  $\in \sigma_{AODV}' \implies vD$  (rt ( $\sigma$  i)) = {}"
  unfolding  $\sigma_{AODV}'$ _def vD_def by simp

lemma oaodv_trans: "trans (opaodv i) = oseqp_sos  $\Gamma_{AODV}$  i"
  by simp

declare
  oseq_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]
  oseq_step_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]

end

```

## 2.9 Global invariant proofs over sequential processes

```

theory B_Global_Invariants
imports B_Seq_Invariants
  B_Aodv_Predicates
  B_Fresher
  B_Quality_Increases
  "../.../AWN/DAWN_Convert"
  B_OAodv
begin

lemma other_quality_increases [elim]:
  assumes "other quality_increases I  $\sigma$   $\sigma'$ "
  shows " $\forall j$ . quality_increases ( $\sigma$  j) ( $\sigma'$  j)"
  using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
  fixes m
  assumes *: "otherwith P I (orecvmsg Q)  $\sigma$   $\sigma'$  a"
    and weakenP: " $\bigwedge \sigma m$ . P  $\sigma$  m  $\implies$  P'  $\sigma$  m"
    and weakenQ: " $\bigwedge \sigma m$ . Q  $\sigma$  m  $\implies$  Q'  $\sigma$  m"
  shows "otherwith P' I (orecvmsg Q')  $\sigma$   $\sigma'$  a"
proof
  fix j
  assume "j  $\notin$  I"
  with * have "P ( $\sigma$  j) ( $\sigma'$  j)" by auto
  thus "P' ( $\sigma$  j) ( $\sigma'$  j)" by (rule weakenP)
next
  from * have "orecvmsg Q  $\sigma$  a" by auto
  thus "orecvmsg Q'  $\sigma$  a"
  by rule (erule weakenQ)
qed

lemma oreceived_msg_inv:
  assumes other: " $\bigwedge \sigma \sigma' m$ . [ $P$   $\sigma$  m; other Q {i}  $\sigma$   $\sigma'$ ]  $\implies$  P  $\sigma'$  m"
    and local: " $\bigwedge \sigma m$ . P  $\sigma$  m  $\implies$  P ( $\sigma$ (i :=  $\sigma$  i(msg := m))) m"
  shows "opaodv i  $\models$  (otherwith Q {i} (orecvmsg P), other Q {i}  $\rightarrow$ )
    onl  $\Gamma_{AODV}$  ( $\lambda(\sigma, l)$ . l  $\in$  {PAodv-:1}  $\rightarrow$  P  $\sigma$  (msg ( $\sigma$  i)))"
proof (inv_cterms, intro impI)
  fix  $\sigma \sigma' l$ 
  assume "l = PAodv-:1  $\rightarrow$  P  $\sigma$  (msg ( $\sigma$  i))"
    and "l = PAodv-:1"
    and "other Q {i}  $\sigma \sigma'$ "
  from this(1-2) have "P  $\sigma$  (msg ( $\sigma$  i))" ..
  hence "P  $\sigma'$  (msg ( $\sigma$  i))" using (other Q {i}  $\sigma \sigma'$ )
    by (rule other)
  moreover from (other Q {i}  $\sigma \sigma'$ ) have " $\sigma'$  i =  $\sigma$  i" ..
  ultimately show "P  $\sigma'$  (msg ( $\sigma'$  i))" by simp
next
  fix  $\sigma \sigma'$  msg
  assume "otherwith Q {i} (orecvmsg P)  $\sigma \sigma'$  (receive msg)"

```

```

    and "σ' i = σ i(msg := msg)"
  from this(1) have "P σ msg"
    and "∀j. j≠i → Q (σ j) (σ' j)" by auto
  from this(1) have "P (σ(i := σ i(msg := msg))) msg" by (rule local)
  thus "P σ' msg"
  proof (rule other)
    from ⟨σ' i = σ i(msg := msg)⟩ and ⟨∀j. j≠i → Q (σ j) (σ' j)⟩
      show "other Q {i} (σ(i := σ i(msg := msg))) σ'"
        by - (rule otherI, auto)
  qed
qed

```

(Equivalent to) Proposition 7.27

lemma local\_quality\_increases:

```

"paadv i ⊨A (recvmsg rreq_rrep_sn →) onll ΓAODV (λ((ξ, _), _, (ξ', _)). quality_increases ξ ξ')"
proof (rule step_invariantI)
  fix s a s'
  assume sr: "s ∈ reachable (paadv i) (recvmsg rreq_rrep_sn)"
    and tr: "(s, a, s') ∈ trans (paadv i)"
    and rm: "recvmsg rreq_rrep_sn a"
  from sr have srTT: "s ∈ reachable (paadv i) TT" ..

  from route_tables_fresher sr tr rm
    have "onll ΓAODV (λ((ξ, _), _, (ξ', _)). ∀dip∈kD (rt ξ). rt ξ ⊆dip rt ξ') (s, a, s')"
      by (rule step_invariantD)

  moreover from known_destinations_increase srTT tr TT_True
    have "onll ΓAODV (λ((ξ, _), _, (ξ', _)). kD (rt ξ) ⊆ kD (rt ξ')) (s, a, s')"
      by (rule step_invariantD)

  moreover from sqns_increase srTT tr TT_True
    have "onll ΓAODV (λ((ξ, _), _, (ξ', _)). ∀ip. sqn (rt ξ) ip ≤ sqn (rt ξ') ip) (s, a, s')"
      by (rule step_invariantD)

  ultimately show "onll ΓAODV (λ((ξ, _), _, (ξ', _)). quality_increases ξ ξ') (s, a, s')"
    unfolding onll_def by auto
qed

```

lemmas olocal\_quality\_increases =

```

open_seq_step_invariant [OF local_quality_increases initiali_aadv oaadv_trans aadv_trans,
  simplified seql_onll_swap]

```

lemma oquality\_increases:

```

"opaadv i ⊨A (otherwith quality_increases {i} (orecvmsg (λ_. rreq_rrep_sn)),
  other quality_increases {i} →)
  onll ΓAODV (λ((σ, _), _, (σ', _)). ∀j. quality_increases (σ j) (σ' j))"
(is "_ ⊨A (?S, _ →) _")
proof (rule onll_ostep_invariantI, simp)
  fix σ p l a σ' p' l'
  assume or: "(σ, p) ∈ oreachable (opaadv i) ?S (other quality_increases {i})"
    and ll: "l ∈ labels ΓAODV p"
    and "?S σ σ' a"
    and tr: "((σ, p), a, (σ', p')) ∈ oseqp_sos ΓAODV i"
    and ll': "l' ∈ labels ΓAODV p'"
  from this(1-3) have "orecvmsg (λ_. rreq_rrep_sn) σ a"
    by (auto dest!: oreachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"
      and QU="other quality_increases {i}"]
      otherwith_actionD)
  with or have orw: "(σ, p) ∈ oreachable (opaadv i) (act (recvmsg rreq_rrep_sn))
    (other quality_increases {i})"
    by - (erule oreachable_weakenE, auto)
  with tr ll ll' and ⟨orecvmsg (λ_. rreq_rrep_sn) σ a⟩ have "quality_increases (σ i) (σ' i)"
    by - (drule onll_ostep_invariantD [OF olocal_quality_increases], auto simp: seql_def)
  with ⟨?S σ σ' a⟩ show "∀j. quality_increases (σ j) (σ' j)"

```

by (auto dest!: otherwith\_syncD)  
qed

lemma rreq\_rrep\_nsqn\_fresh\_any\_step\_invariant:

"opaadv i  $\models_A$  (act (recvmsg rreq\_rrep\_sn), other A {i}  $\rightarrow$ )  
onll  $\Gamma_{AODV}$  ( $\lambda((\sigma, \_), a, \_).$  anycast (msg\_fresh  $\sigma$ ) a)"

proof (rule ostep\_invariantI, simp del: act\_simp)

fix  $\sigma$  p a  $\sigma'$  p'

assume or: " $(\sigma, p) \in$  oreachable (opaadv i) (act (recvmsg rreq\_rrep\_sn)) (other A {i})"  
and " $((\sigma, p), a, (\sigma', p')) \in$  oseqp\_sos  $\Gamma_{AODV}$  i"  
and recv: "act (recvmsg rreq\_rrep\_sn)  $\sigma$   $\sigma'$  a"

obtain l l' where " $l \in$  labels  $\Gamma_{AODV}$  p" and " $l' \in$  labels  $\Gamma_{AODV}$  p'"  
by (metis aadv\_ex\_label)

from  $\langle((\sigma, p), a, (\sigma', p')) \in$  oseqp\_sos  $\Gamma_{AODV}$  i

have tr: " $((\sigma, p), a, (\sigma', p')) \in$  trans (opaadv i)" by simp

have "anycast (rreq\_rrep\_fresh (rt ( $\sigma$  i))) a"

proof -

have "opaadv i  $\models_A$  (act (recvmsg rreq\_rrep\_sn), other A {i}  $\rightarrow$ )  
onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_).$  anycast (rreq\_rrep\_fresh (rt  $\xi$ )) a))"

by (rule ostep\_invariant\_weakenE [OF

open\_seq\_step\_invariant [OF rreq\_rrep\_fresh\_any\_step\_invariant initiali\_aadv,  
simplified seqll\_onll\_swap]]) auto

hence "onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_).$  anycast (rreq\_rrep\_fresh (rt  $\xi$ )) a))  
 $((\sigma, p), a, (\sigma', p'))$ "

using or tr recv by - (erule(4) ostep\_invariantE)

thus ?thesis

using  $\langle l \in$  labels  $\Gamma_{AODV}$  p  $\rangle$  and  $\langle l' \in$  labels  $\Gamma_{AODV}$  p'  $\rangle$  by auto

qed

moreover have "anycast (rerr\_invalid (rt ( $\sigma$  i))) a"

proof -

have "opaadv i  $\models_A$  (act (recvmsg rreq\_rrep\_sn), other A {i}  $\rightarrow$ )  
onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_).$  anycast (rerr\_invalid (rt  $\xi$ )) a))"

by (rule ostep\_invariant\_weakenE [OF

open\_seq\_step\_invariant [OF rerr\_invalid\_any\_step\_invariant initiali\_aadv,  
simplified seqll\_onll\_swap]]) auto

hence "onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_).$  anycast (rerr\_invalid (rt  $\xi$ )) a))  
 $((\sigma, p), a, (\sigma', p'))$ "

using or tr recv by - (erule(4) ostep\_invariantE)

thus ?thesis

using  $\langle l \in$  labels  $\Gamma_{AODV}$  p  $\rangle$  and  $\langle l' \in$  labels  $\Gamma_{AODV}$  p'  $\rangle$  by auto

qed

moreover have "anycast rreq\_rrep\_sn a"

proof -

from or tr recv

have "onll  $\Gamma_{AODV}$  (seqll i ( $\lambda(\_, a, \_).$  anycast rreq\_rrep\_sn a))  $((\sigma, p), a, (\sigma', p'))$ "

by (rule ostep\_invariantE [OF

open\_seq\_step\_invariant [OF rreq\_rrep\_sn\_any\_step\_invariant initiali\_aadv  
oaadv\_trans aadv\_trans,  
simplified seqll\_onll\_swap]])

thus ?thesis

using  $\langle l \in$  labels  $\Gamma_{AODV}$  p  $\rangle$  and  $\langle l' \in$  labels  $\Gamma_{AODV}$  p'  $\rangle$  by auto

qed

moreover have "anycast ( $\lambda m.$  not\_Pkt m  $\rightarrow$  msg\_sender m = i) a"

proof -

have "opaadv i  $\models_A$  (act (recvmsg rreq\_rrep\_sn), other A {i}  $\rightarrow$ )

onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_).$  anycast ( $\lambda m.$  not\_Pkt m  $\rightarrow$  msg\_sender m = i) a))"

by (rule ostep\_invariant\_weakenE [OF

open\_seq\_step\_invariant [OF sender\_ip\_valid initiali\_aadv,  
simplified seqll\_onll\_swap]]) auto

thus ?thesis using or tr recv  $\langle l \in$  labels  $\Gamma_{AODV}$  p  $\rangle$  and  $\langle l' \in$  labels  $\Gamma_{AODV}$  p'  $\rangle$



```

      by - (drule(3) onll_ostep_invariantD, auto)
qed

ultimately have "anycast (msg_fresh  $\sigma$ ) a"
  by (simp_all add: anycast_def
      del: msg_fresh
      split: seq_action.split_asm msg.split_asm) simp_all
thus "onll  $\Gamma_{AODV} (\lambda((\sigma, \_), a, \_). \text{anycast } (\text{msg\_fresh } \sigma) a) ((\sigma, p), a, (\sigma', p'))"$ "
  by auto
qed

lemma oreceived_rreq_rrep_nsqn_fresh_inv:
"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ 
  onl  $\Gamma_{AODV} (\lambda(\sigma, l). l \in \{PAodv-:1\} \rightarrow \text{msg\_fresh } \sigma (\text{msg } (\sigma i)))"$ )"
proof (rule oreceived_msg_inv)
  fix  $\sigma \sigma' m$ 
  assume *: "msg_fresh  $\sigma m$ "
  and "other quality_increases {i}  $\sigma \sigma'$ "
  from this(2) have " $\forall j. \text{quality\_increases } (\sigma j) (\sigma' j)" ..$ 
  thus "msg_fresh  $\sigma' m$ " using * ..
next
  fix  $\sigma m$ 
  assume "msg_fresh  $\sigma m$ "
  thus "msg_fresh ( $\sigma(i := \sigma i(\text{msg} := m))$ ) m"
proof (cases m)
  fix dests sip
  assume "m = Rerr dests sip"
  with (msg_fresh  $\sigma m$ ) show ?thesis by auto
qed auto
qed

lemma oquality_increases_nsqn_fresh:
"opaadv i  $\models_A$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ 
  onll  $\Gamma_{AODV} (\lambda((\sigma, \_), \_, (\sigma', \_)). \forall j. \text{quality\_increases } (\sigma j) (\sigma' j))"$ )"
by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ 
  onl  $\Gamma_{AODV} (\text{seq1 } i (\lambda(\xi, l). l \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-:n \mid n. \text{True}\} \rightarrow 1 \leq \text{osn } \xi))"$ )"
by (rule oinvariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_aadv]])
(auto simp: seq1_onl_swap)

lemma rreq_sip:
"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ 
  onl  $\Gamma_{AODV} (\lambda(\sigma, l).
    (l \in \{PAodv-:4, PAodv-:5, PRreq-:0, PRreq-:2\} \wedge \text{sip } (\sigma i) \neq \text{oip } (\sigma i))
    \rightarrow \text{oip } (\sigma i) \in \text{kD}(\text{rt } (\sigma (\text{sip } (\sigma i))))
    \wedge \text{nsqn } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i)) \geq \text{osn } (\sigma i)
    \wedge (\text{nsqn } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i)) = \text{osn } (\sigma i)
    \rightarrow (\text{hops } (\sigma i) \geq \text{the } (\text{dhops } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i)))
    \vee \text{the } (\text{flag } (\text{rt } (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i))) = \text{inv}))"$ )"
(is " $\_ \models (?S, ?U \rightarrow) \_$ ")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh
  aadv_wf oaadv_trans]
  onl_oinvariant_sterms [OF aadv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
  onl_oinvariant_sterms [OF aadv_wf oosn_rreq]
  simp add: seq1simp
  simp del: One_nat_def, rule impI)
  fix  $\sigma \sigma' p l$ 
  assume " $(\sigma, p) \in \text{oreachable } (\text{opaadv } i) ?S ?U"$ "

```

```

and "l ∈ labels ΓAODV p"
and pre:
  "(l = PAadv-:4 ∨ l = PAadv-:5 ∨ l = PRreq-:0 ∨ l = PRreq-:2) ∧ sip (σ i) ≠ oip (σ i)
  → oip (σ i) ∈ kD (rt (σ (sip (σ i))))
  ∧ osn (σ i) ≤ nsqn (rt (σ (sip (σ i)))) (oip (σ i))
  ∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ i)) = osn (σ i)
  → the (dhops (rt (σ (sip (σ i)))) (oip (σ i))) ≤ hops (σ i)
  ∨ the (flag (rt (σ (sip (σ i)))) (oip (σ i))) = inv)"
and "other quality_increases {i} σ σ'"
and hyp: "(l=PAadv-:4 ∨ l=PAadv-:5 ∨ l=PRreq-:0 ∨ l=PRreq-:2) ∧ sip (σ' i) ≠ oip (σ' i)"
(is "?labels ∧ sip (σ' i) ≠ oip (σ' i)")
from this(4) have "σ' i = σ i" ..
with hyp have hyp': "?labels ∧ sip (σ i) ≠ oip (σ i)" by simp
show "oip (σ' i) ∈ kD (rt (σ' (sip (σ' i))))
  ∧ osn (σ' i) ≤ nsqn (rt (σ' (sip (σ' i)))) (oip (σ' i))
  ∧ (nsqn (rt (σ' (sip (σ' i)))) (oip (σ' i)) = osn (σ' i)
  → the (dhops (rt (σ' (sip (σ' i)))) (oip (σ' i))) ≤ hops (σ' i)
  ∨ the (flag (rt (σ' (sip (σ' i)))) (oip (σ' i))) = inv)"
proof (cases "sip (σ i) = i")
assume "sip (σ i) ≠ i"
from ⟨other quality_increases {i} σ σ'⟩
have "quality_increases (σ (sip (σ i))) (σ' (sip (σ' i)))"
by (rule otherE) (clarsimp simp: ⟨sip (σ i) ≠ i⟩)
moreover from ⟨(σ, p) ∈ oreachable (opaadv i) ?S ?U⟩ ⟨l ∈ labels ΓAODV p⟩ and hyp
have "l ≤ osn (σ' i)"
by (auto dest!: onl_oinvariant_weakenD [OF oosn_rreq]
simp add: seqlsimp ⟨σ' i = σ i⟩)
moreover from ⟨sip (σ i) ≠ i⟩ hyp' and pre
have "oip (σ' i) ∈ kD (rt (σ (sip (σ i))))
  ∧ osn (σ' i) ≤ nsqn (rt (σ (sip (σ i)))) (oip (σ' i))
  ∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ' i)) = osn (σ' i)
  → the (dhops (rt (σ (sip (σ i)))) (oip (σ' i))) ≤ hops (σ' i)
  ∨ the (flag (rt (σ (sip (σ i)))) (oip (σ' i))) = inv)"
by (auto simp: ⟨σ' i = σ i⟩)
ultimately show ?thesis
by (rule quality_increases_rreq_rrep_props)
next
assume "sip (σ i) = i" thus ?thesis
using ⟨σ' i = σ i⟩ hyp and pre by auto
qed
qed (auto elim!: quality_increases_rreq_rrep_props')

```

lemma odsn\_rrep:

```

"opaadv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i} →)
  onl ΓAODV (seq1 i (λ(ξ, l). l ∈ {PAadv-:6, PAadv-:7} ∪ {PRrep-:n/n. True} → l ≤ dsn ξ))"
by (rule oinvariant_weakenE [OF open_seq_invariant [OF dsn_rrep initiali_adv]])
(auto simp: seq1_onl_swap)

```

lemma rrep\_sip:

```

"opaadv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i} →)
  onl ΓAODV (λ(σ, l).
  (l ∈ {PAadv-:6, PAadv-:7, PRrep-:0, PRrep-:1} ∧ sip (σ i) ≠ dip (σ i)
  → dip (σ i) ∈ kD(rt (σ (sip (σ i))))
  ∧ nsqn (rt (σ (sip (σ i)))) (dip (σ i)) ≥ dsn (σ i)
  ∧ (nsqn (rt (σ (sip (σ i)))) (dip (σ i)) = dsn (σ i)
  → (hops (σ i) ≥ the (dhops (rt (σ (sip (σ i)))) (dip (σ i)))
  ∨ the (flag (rt (σ (sip (σ i)))) (dip (σ i))) = inv)))"
(is "_ ⊨ (?S, ?U →) _")

```

```

proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh adv_wf
  oaadv_trans]
  onl_oinvariant_sterms [OF adv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
  onl_oinvariant_sterms [OF adv_wf odsn_rrep]

```

```

      simp del: One_nat_def, rule impI)
fix  $\sigma \sigma' p l$ 
assume "( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U"
  and " $l \in$  labels  $\Gamma_{AODV} p$ "
  and pre:
    "( $l = PAadv-:6 \vee l = PAadv-:7 \vee l = PRrep-:0 \vee l = PRrep-:1$ )  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i)
     $\rightarrow$  dip ( $\sigma$  i)  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
       $\wedge$  dsn ( $\sigma$  i)  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i)) = dsn ( $\sigma$  i)
         $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i)))  $\leq$  hops ( $\sigma$  i)
           $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = inv)"
  and "other quality_increases {i}  $\sigma \sigma'$ "
  and hyp: "( $l=PAadv-:6 \vee l=PAadv-:7 \vee l=PRrep-:0 \vee l=PRrep-:1$ )  $\wedge$  sip ( $\sigma'$  i)  $\neq$  dip ( $\sigma'$  i)"
    (is "?labels  $\wedge$  sip ( $\sigma'$  i)  $\neq$  dip ( $\sigma'$  i)")
from this(4) have " $\sigma' i = \sigma i$ " ..
with hyp have hyp': "?labels  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i)" by simp
show "dip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma'$  (sip ( $\sigma' i$ ))))
   $\wedge$  dsn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ ))
   $\wedge$  (nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ )) = dsn ( $\sigma' i$ )
     $\rightarrow$  the (dhops (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
       $\vee$  the (flag (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ ))) = inv)"
proof (cases "sip ( $\sigma$  i) = i")
  assume "sip ( $\sigma$  i)  $\neq$  i"
  from (other quality_increases {i}  $\sigma \sigma'$ )
    have "quality_increases ( $\sigma$  (sip ( $\sigma$  i))) ( $\sigma'$  (sip ( $\sigma' i$ )))"
      by (rule otherE) (clarsimp simp: (sip ( $\sigma$  i)  $\neq$  i))
  moreover from (( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U) (l  $\in$  labels  $\Gamma_{AODV} p$ ) and hyp
    have " $l \leq$  dsn ( $\sigma' i$ )"
      by (auto dest!: onl_oinvariant_weakenD [OF odsn_rrep]
          simp add: seqsimp (sigma' i = sigma i))
  moreover from (sip ( $\sigma$  i)  $\neq$  i) hyp' and pre
    have "dip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
       $\wedge$  dsn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ ))
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ )) = dsn ( $\sigma' i$ )
         $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
           $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ ))) = inv)"
      by (auto simp: (sigma' i = sigma i))
  ultimately show ?thesis
    by (rule quality_increases_rreq_rrep_props)
next
  assume "sip ( $\sigma$  i) = i" thus ?thesis
    using (sigma' i = sigma i) hyp and pre by auto
qed
qed (auto simp add: seqsimp elim!: quality_increases_rreq_rrep_props')

```

```

lemma rerr_sip:
  "opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i}  $\rightarrow$ )
    onl  $\Gamma_{AODV} (\lambda(\sigma, l).$ 
       $l \in \{PAadv-:8, PAadv-:9, PRerr-:0, PRerr-:1\}$ 
       $\rightarrow (\forall rip \in \text{dom}(\text{dests } (\sigma \text{ i})). rip \in \text{kD}(\text{rt } (\sigma \text{ (sip } (\sigma \text{ i})))) \wedge$ 
        the (dests ( $\sigma$  i) rip) - 1  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) rip))"
  (is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")
proof -
  { fix dests rip sip rsn and  $\sigma \sigma' ::$  "ip  $\Rightarrow$  state"
    assume qinc: " $\forall j. \text{quality\_increases } (\sigma j) (\sigma' j)$ "
      and *: " $\forall rip \in \text{dom } \text{dests}. rip \in \text{kD} (\text{rt } (\sigma \text{ sip}))$ 
         $\wedge$  the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip"
      and "dests rip = Some rsn"
    from this(3) have "rip  $\in$  dom dests" by auto
    with * and (dests rip = Some rsn) have "rip  $\in$  kD (rt ( $\sigma$  sip))"
      and "rsn - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip"
    by (auto dest!: bspec)
    from qinc have "quality_increases ( $\sigma$  sip) ( $\sigma'$  sip)" ..
  }

```

```

have "rip ∈ kD(rt (σ' sip)) ∧ rsn - 1 ≤ nsqn (rt (σ' sip)) rip"
proof
  from ⟨rip ∈ kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  show "rip ∈ kD(rt (σ' sip))" ..
next
  from ⟨rip ∈ kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  have "nsqn (rt (σ sip)) rip ≤ nsqn (rt (σ' sip)) rip" ..
  with ⟨rsn - 1 ≤ nsqn (rt (σ sip)) rip⟩ show "rsn - 1 ≤ nsqn (rt (σ' sip)) rip"
  by (rule le_trans)
qed
} note partial = this

show ?thesis
by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
  oadv_trans]
  onl_oinvariant_sterms [OF aadv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
  other_quality_increases other_localD
  simp del: One_nat_def, intro conjI]
  (clarsimp simp del: One_nat_def split: split_if_asm option.split_asm, erule(2) partial)+
qed

lemma prerr_guard: "paadv i ⊨
  onl ΓAODV (λ(ξ, l). (l = PRerr-:1
    → (∀ ip ∈ dom(dests ξ). ip ∈ vD(rt ξ)
      ∧ the (nhop (rt ξ) ip) = sip ξ
      ∧ sqn (rt ξ) ip < the (dests ξ ip))))"
by (inv_cterms) (clarsimp split: option.split_asm split_if_asm)

lemmas oadpreRT_welldefined =
  open_seq_invariant [OF oadpreRT_welldefined initiali_aadv oaadv_trans aadv_trans,
    simplified seq_l_onl_swap,
    THEN oinvariant_anyact]

lemmas odests_vD_inc_sqn =
  open_seq_invariant [OF odests_vD_inc_sqn initiali_aadv oaadv_trans aadv_trans,
    simplified seq_l_onl_swap,
    THEN oinvariant_anyact]

lemmas oprerr_guard =
  open_seq_invariant [OF prerr_guard initiali_aadv oaadv_trans aadv_trans,
    simplified seq_l_onl_swap,
    THEN oinvariant_anyact]

Proposition 7.28

lemma seq_compare_next_hop':
  "opaadv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} → onl ΓAODV (λ(σ, _).
      ∀ dip. let nhop = the (nhop (rt (σ i)) dip)
        in dip ∈ kD(rt (σ i)) ∧ nhop ≠ dip →
          dip ∈ kD(rt (σ nhop)) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhop)) dip)"
(is "_ ⊨ (?S, ?U →) _")
proof -
{ fix nhop and σ σ' :: "ip ⇒ state"
  assume pre: "∀ dip ∈ kD(rt (σ i)). nhop dip ≠ dip →
    dip ∈ kD(rt (σ (nhop dip))) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
  and qinc: "∀ j. quality_increases (σ j) (σ' j)"
  have "∀ dip ∈ kD(rt (σ i)). nhop dip ≠ dip →
    dip ∈ kD(rt (σ' (nhop dip))) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
  proof (intro ballI impI)
    fix dip
    assume "dip ∈ kD(rt (σ i))"
    and "nhop dip ≠ dip"
    with pre have "dip ∈ kD(rt (σ (nhop dip)))"

```

```

    and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
  by auto
from qinc have qinc_nhop: "quality_increases (σ (nhop dip)) (σ' (nhop dip))" ..
with ⟨dip∈kD(rt (σ (nhop dip)))⟩ have "dip∈kD (rt (σ' (nhop dip)))" ..

moreover have "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
proof -
  from ⟨dip∈kD(rt (σ (nhop dip)))⟩ qinc_nhop
  have "nsqn (rt (σ (nhop dip))) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ..
  with ⟨nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip⟩ show ?thesis
  by simp
qed

ultimately show "dip∈kD(rt (σ' (nhop dip)))
  ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ..
qed
} note basic = this

{ fix nhop and σ σ' :: "ip ⇒ state"
  assume pre: "∀dip∈kD(rt (σ i)). nhop dip ≠ dip → dip∈kD(rt (σ (nhop dip)))
    ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
  and ndest: "∀ripc∈dom (dests (σ i)). ripc ∈ kD (rt (σ (sip (σ i))))
    ∧ the (dests (σ i) ripc) - 1 ≤ nsqn (rt (σ (sip (σ i)))) ripc"
  and issip: "∀ip∈dom (dests (σ i)). nhop ip = sip (σ i)"
  and qinc: "∀j. quality_increases (σ j) (σ' j)"
  have "∀dip∈kD(rt (σ i)). nhop dip ≠ dip → dip ∈ kD (rt (σ' (nhop dip)))
    ∧ nsqn (invalidate (rt (σ i)) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
  proof (intro ballI impI)
    fix dip
    assume "dip∈kD(rt (σ i))"
    and "nhop dip ≠ dip"
    with pre and qinc have "dip∈kD(rt (σ' (nhop dip)))"
    and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
    by (auto dest!: basic)

    have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
    proof (cases "dip∈dom (dests (σ i))")
      assume "dip∈dom (dests (σ i))"
      with ⟨dip∈kD(rt (σ i))⟩ obtain dsn where "dests (σ i) dip = Some dsn"
      by auto
      with ⟨dip∈kD(rt (σ i))⟩ have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = dsn - 1"
      by (rule nsqn_invalidate_eq)
      moreover have "dsn - 1 ≤ nsqn (rt (σ' (nhop dip))) dip"
      proof -
        from ⟨dests (σ i) dip = Some dsn⟩ have "the (dests (σ i) dip) = dsn" by simp
        with ndest and ⟨dip∈dom (dests (σ i))⟩ have "dip ∈ kD (rt (σ (sip (σ i))))"
        "dsn - 1 ≤ nsqn (rt (σ (sip (σ i)))) dip"
        by auto
        moreover from issip and ⟨dip∈dom (dests (σ i))⟩ have "nhop dip = sip (σ i)" ..
        ultimately have "dip ∈ kD (rt (σ (nhop dip)))"
        and "dsn - 1 ≤ nsqn (rt (σ (nhop dip))) dip" by auto
        with qinc show "dsn - 1 ≤ nsqn (rt (σ' (nhop dip))) dip"
        by simp (metis kD_nsqn_quality_increases_trans)
      qed
      ultimately show ?thesis by simp
    next
    assume "dip ∉ dom (dests (σ i))"
    with ⟨dip∈kD(rt (σ i))⟩
    have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = nsqn (rt (σ i)) dip"
    by (rule nsqn_invalidate_other)
    with ⟨nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip⟩ show ?thesis by simp
  qed
  with ⟨dip∈kD(rt (σ' (nhop dip)))⟩
  show "dip ∈ kD (rt (σ' (nhop dip)))"
}

```

```

       $\wedge$  nsqn (invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i))) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip" ..
qed
} note basic_prerr = this

{ fix  $\sigma$   $\sigma'$  :: "ip  $\Rightarrow$  state"
  assume a1: " $\forall$ dip $\in$ kD(rt ( $\sigma$  i)). the (nhop (rt ( $\sigma$  i)) dip)  $\neq$  dip
     $\rightarrow$  dip $\in$ kD(rt ( $\sigma$  (the (nhop (rt ( $\sigma$  i)) dip))))
       $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma$  (the (nhop (rt ( $\sigma$  i)) dip)))) dip"
  and a2: " $\forall$ j. quality_increases ( $\sigma$  j) ( $\sigma'$  j)"
  have " $\forall$ dip $\in$ kD(rt ( $\sigma$  i)).
    the (nhop (update (rt ( $\sigma$  i)) (sip ( $\sigma$  i)) (0, unk, val, Suc 0, sip ( $\sigma$  i), { }))) dip)  $\neq$  dip  $\rightarrow$ 
    dip $\in$ kD(rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) (sip ( $\sigma$  i))
      (0, unk, val, Suc 0, sip ( $\sigma$  i), { })))
        dip))))  $\wedge$ 
    nsqn (update (rt ( $\sigma$  i)) (sip ( $\sigma$  i)) (0, unk, val, Suc 0, sip ( $\sigma$  i), { }))) dip
     $\leq$  nsqn (rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) (sip ( $\sigma$  i))
      (0, unk, val, Suc 0, sip ( $\sigma$  i), { })))
        dip))))
    dip" (is " $\forall$ dip $\in$ kD(rt ( $\sigma$  i)). ?P dip")
proof
  fix dip
  assume "dip $\in$ kD(rt ( $\sigma$  i))"
  with a1 and a2
    have "the (nhop (rt ( $\sigma$  i)) dip)  $\neq$  dip  $\rightarrow$  dip $\in$ kD(rt ( $\sigma'$  (the (nhop (rt ( $\sigma$  i)) dip))))
       $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma'$  (the (nhop (rt ( $\sigma$  i)) dip)))) dip"
    by - (drule(1) basic, auto)
  thus "?P dip" by (cases "dip = sip ( $\sigma$  i)") auto
qed
} note nhop_update_sip = this

{ fix  $\sigma$   $\sigma'$  oip sip osn hops
  assume pre: " $\forall$ dip $\in$ kD (rt ( $\sigma$  i)). the (nhop (rt ( $\sigma$  i)) dip)  $\neq$  dip
     $\rightarrow$  dip $\in$ kD(rt ( $\sigma$  (the (nhop (rt ( $\sigma$  i)) dip))))
       $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma$  (the (nhop (rt ( $\sigma$  i)) dip)))) dip"
  and qinc: " $\forall$ j. quality_increases ( $\sigma$  j) ( $\sigma'$  j)"
  and *: "sip  $\neq$  oip  $\rightarrow$  oip $\in$ kD(rt ( $\sigma$  sip))
     $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma$  sip)) oip
     $\wedge$  (nsqn (rt ( $\sigma$  sip)) oip = osn
       $\rightarrow$  the (dhops (rt ( $\sigma$  sip)) oip)  $\leq$  hops
       $\vee$  the (flag (rt ( $\sigma$  sip)) oip) = inv)"
  from pre and qinc
    have pre': " $\forall$ dip $\in$ kD (rt ( $\sigma$  i)). the (nhop (rt ( $\sigma$  i)) dip)  $\neq$  dip
       $\rightarrow$  dip $\in$ kD(rt ( $\sigma'$  (the (nhop (rt ( $\sigma$  i)) dip))))
         $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma'$  (the (nhop (rt ( $\sigma$  i)) dip)))) dip"
    by (rule basic)
  have "(the (nhop (update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { }))) oip)  $\neq$  oip
     $\rightarrow$  oip $\in$ kD(rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) oip
      (osn, kno, val, Suc hops, sip, { }))) oip))))
       $\wedge$  nsqn (update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { }))) oip
       $\leq$  nsqn (rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) oip
        (osn, kno, val, Suc hops, sip, { }))) oip)))) oip)"
    (is "?nhop_not_oip  $\rightarrow$  ?oip_in_kD  $\wedge$  ?nsqn_le_nsqn")
  proof (rule, split update_rt_split_asm)
    assume "rt ( $\sigma$  i) = update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { })"
    and "the (nhop (rt ( $\sigma$  i)) oip)  $\neq$  oip"
    with pre' show "?oip_in_kD  $\wedge$  ?nsqn_le_nsqn" by auto
  next
    assume rtnot: "rt ( $\sigma$  i)  $\neq$  update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { })"
    and notoip: ?nhop_not_oip
    with * qinc have ?oip_in_kD
      by (clarsimp elim!: kD_quality_increases)
    moreover with * pre qinc rtnot notoip have ?nsqn_le_nsqn
      by simp (metis kD_nsqn_quality_increases_trans)
    ultimately show "?oip_in_kD  $\wedge$  ?nsqn_le_nsqn" ..

```

```

qed
} note update1 = this

{ fix  $\sigma$   $\sigma'$  oip sip osn hops
  assume pre: " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}$ 
     $\rightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$ 
     $\wedge \text{nsqn}(\text{rt}(\sigma i)) \text{dip} \leq \text{nsqn}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))) \text{dip}"$ 
  and qinc: " $\forall j. \text{quality\_increases}(\sigma j)(\sigma' j)"$ 
  and *: " $\text{sip} \neq \text{oip} \rightarrow \text{oip} \in \text{kD}(\text{rt}(\sigma \text{sip}))$ 
     $\wedge \text{osn} \leq \text{nsqn}(\text{rt}(\sigma \text{sip})) \text{oip}$ 
     $\wedge (\text{nsqn}(\text{rt}(\sigma \text{sip})) \text{oip} = \text{osn})$ 
     $\rightarrow \text{the}(\text{dhops}(\text{rt}(\sigma \text{sip})) \text{oip}) \leq \text{hops}$ 
     $\vee \text{the}(\text{flag}(\text{rt}(\sigma \text{sip})) \text{oip}) = \text{inv}"$ 

  from pre and qinc
  have pre': " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}$ 
     $\rightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$ 
     $\wedge \text{nsqn}(\text{rt}(\sigma i)) \text{dip} \leq \text{nsqn}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))) \text{dip}"$ 
  by (rule basic)
  have " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)).$ 
     $\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip}, \{\}) \text{dip})) \neq \text{dip}$ 
     $\rightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}$ 
       $(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip}, \{\}) \text{dip}))))$ 
     $\wedge \text{nsqn}(\text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip}, \{\}) \text{dip})$ 
     $\leq \text{nsqn}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}$ 
       $(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip}, \{\}) \text{dip})))) \text{dip}"$ 
    (is " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \_ \rightarrow ?\text{dip\_in\_kD} \text{dip} \wedge ?\text{nsqn\_le\_nsqn} \text{dip}"$ )
  proof (intro ballI impI, split update_rt_split_asm)
  fix dip
  assume "dip  $\in$  kD(rt( $\sigma$  i))"
  and "the (nhop (rt ( $\sigma$  i)) dip)  $\neq$  dip"
  and "rt ( $\sigma$  i) = update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, {})"
  with pre' show "?dip_in_kD dip  $\wedge$  ?nsqn_le_nsqn dip" by simp
  next
  fix dip
  assume "dip  $\in$  kD(rt( $\sigma$  i))"
  and notdip: "the (nhop (update (rt ( $\sigma$  i)) oip
    (osn, kno, val, Suc hops, sip, {})) dip)  $\neq$  dip"
  and rtnot: "rt ( $\sigma$  i)  $\neq$  update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, {})"
  show "?dip_in_kD dip  $\wedge$  ?nsqn_le_nsqn dip"
  proof (cases "dip = oip")
  assume "dip  $\neq$  oip"
  with pre' (dip  $\in$  kD(rt( $\sigma$  i))) notdip
  show ?thesis by clarsimp
  next
  assume "dip = oip"
  with rtnot qinc (dip  $\in$  kD(rt( $\sigma$  i))) notdip *
  have "?dip_in_kD dip"
  by simp (metis kD_quality_increases)
  moreover from (dip = oip) rtnot qinc (dip  $\in$  kD(rt( $\sigma$  i))) notdip *
  have "?nsqn_le_nsqn dip" by simp (metis kD_nsqn_quality_increases_trans)
  ultimately show ?thesis ..
  qed
  qed
} note update2 = this

have "opaadv i  $\models$  ( $?S, ?U \rightarrow$ ) onl  $\Gamma_{AODV}(\lambda(\sigma, \_).$ 
   $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}$ 
   $\rightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$ 
   $\wedge \text{nsqn}(\text{rt}(\sigma i)) \text{dip} \leq \text{nsqn}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))) \text{dip}"$ 
  by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
    oadv_trans]
    onl_oInvariant_sterms [OF aadv_wf oaddpreRT_welldefined]
    onl_oInvariant_sterms [OF aadv_wf odests_vD_inc_nsqn]
    onl_oInvariant_sterms [OF aadv_wf oprerr_guard]

```

```

        onl_oinvariant_sterms [OF aadv_wf rreq_sip]
        onl_oinvariant_sterms [OF aadv_wf rrep_sip]
        onl_oinvariant_sterms [OF aadv_wf rerr_sip]
        other_quality_increases
        other_localD
    solve: basic basic_prerr
    simp add: seqlsimp nsqn_invalidate nhop_update_sip
    simp del: One_nat_def
(rule conjI, erule(2) update1, erule(2) update2)+

```

thus ?thesis unfolding Let\_def by auto  
qed

Proposition 7.30

```

lemmas okD_unk_or_atleast_one =
  open_seq_invariant [OF kD_unk_or_atleast_one initiali_aadv,
    simplified seql_onl_swap]

```

```

lemmas ozero_seq_unk_hops_one =
  open_seq_invariant [OF zero_seq_unk_hops_one initiali_aadv,
    simplified seql_onl_swap]

```

lemma oreachable\_fresh\_okD\_unk\_or\_atleast\_one:

```

  fixes dip
  assumes "(σ, p) ∈ oreachable (opaadv i)
    (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
      ∧ msg_zhops m)))
    (other quality_increases {i})"
  and "dip ∈ kD(rt (σ i))"
  shows "π3(the (rt (σ i) dip)) = unk ∨ 1 ≤ π2(the (rt (σ i) dip))"
  (is "?P dip")
proof -
  have "∃l. l ∈ labels ΓAADV p" by (metis aadv_ex_label)
  with assms(1) have "∀dip ∈ kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaadv_trans aadv_trans]],
    auto dest!: otherwith_actionD onlD simp: seqlsimp)
  with ⟨dip ∈ kD(rt (σ i))⟩ show ?thesis by simp
qed

```

lemma oreachable\_fresh\_ozero\_seq\_unk\_hops\_one:

```

  fixes dip
  assumes "(σ, p) ∈ oreachable (opaadv i)
    (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
      ∧ msg_zhops m)))
    (other quality_increases {i})"
  and "dip ∈ kD(rt (σ i))"
  shows "sqn (rt (σ i) dip) = 0 →
    sqnf (rt (σ i) dip) = unk
    ∧ the (dhops (rt (σ i) dip)) = 1
    ∧ the (nhop (rt (σ i) dip)) = dip"
  (is "?P dip")
proof -
  have "∃l. l ∈ labels ΓAADV p" by (metis aadv_ex_label)
  with assms(1) have "∀dip ∈ kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaadv_trans aadv_trans]],
    auto dest!: onlD otherwith_actionD simp: seqlsimp)
  with ⟨dip ∈ kD(rt (σ i))⟩ show ?thesis by simp
qed

```

lemma seq\_nhop\_quality\_increases':

```

shows "opaadv i ⊨ (otherwith (op=) {i}
  (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
  other quality_increases {i} →)
  onl ΓAADV (λ(σ, _). ∀dip. let nhip = the (nhop (rt (σ i) dip))

```



$$\begin{aligned} & \text{in } \text{dip} \in \text{vD}(\text{rt}(\sigma i)) \cap \text{vD}(\text{rt}(\sigma \text{nhip})) \\ & \quad \wedge \text{nhip} \neq \text{dip} \\ & \quad \rightarrow (\text{rt}(\sigma i)) \sqsubseteq_{\text{dip}} (\text{rt}(\sigma \text{nhip})) \end{aligned}$$

(is " $\_ \models$  (?S i,  $\_ \rightarrow$ )  $\_$ ")

proof -

have weaken:

" $\bigwedge P I Q R P. p \models$  (otherwith quality\_increases I (orecvmsg Q), other quality\_increases I  $\rightarrow$ ) P  
 $\implies p \models$  (otherwith (op=) I (orecvmsg ( $\lambda \sigma m. Q \sigma m \wedge R \sigma m$ )), other quality\_increases I  $\rightarrow$ ) P"  
 by auto

{

fix i a and  $\sigma \sigma' ::$  "ip  $\Rightarrow$  state"

assume a1: " $\forall \text{dip}. \text{dip} \in \text{vD}(\text{rt}(\sigma i))$   
 $\wedge \text{dip} \in \text{vD}(\text{rt}(\sigma (\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$   
 $\wedge (\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})) \neq \text{dip}$   
 $\rightarrow \text{rt}(\sigma i) \sqsubseteq_{\text{dip}} \text{rt}(\sigma (\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))"$

and ow: "?S i  $\sigma \sigma'$  a"

have " $\forall \text{dip}. \text{dip} \in \text{vD}(\text{rt}(\sigma i))$   
 $\wedge \text{dip} \in \text{vD}(\text{rt}(\sigma' (\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$   
 $\wedge (\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})) \neq \text{dip}$   
 $\rightarrow \text{rt}(\sigma i) \sqsubseteq_{\text{dip}} \text{rt}(\sigma' (\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))"$

proof clarify

fix dip

assume a2: "dip  $\in$  vD(rt( $\sigma i$ ))"  
 and a3: "dip  $\in$  vD(rt( $\sigma'$  (the (nhop (rt ( $\sigma i$ )) dip))))"  
 and a4: "(the (nhop (rt ( $\sigma i$ )) dip))  $\neq$  dip"

from ow have " $\forall j. j \neq i \rightarrow \sigma j = \sigma' j$ " by auto

show "rt( $\sigma i$ )  $\sqsubseteq_{\text{dip}}$  rt( $\sigma'$  (the (nhop (rt ( $\sigma i$ )) dip)))"

proof (cases "(the (nhop (rt ( $\sigma i$ )) dip)) = i")

assume "(the (nhop (rt ( $\sigma i$ )) dip)) = i"  
 with  $\langle \text{dip} \in \text{vD}(\text{rt}(\sigma i)) \rangle$  have "dip  $\in$  vD(rt( $\sigma$  (the (nhop (rt ( $\sigma i$ )) dip))))" by simp  
 with a1 a2 a4 have "rt( $\sigma i$ )  $\sqsubseteq_{\text{dip}}$  rt( $\sigma$  (the (nhop (rt ( $\sigma i$ )) dip)))" by simp  
 with  $\langle (\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})) = i \rangle$  have "rt( $\sigma i$ )  $\sqsubseteq_{\text{dip}}$  rt( $\sigma i$ )" by simp  
 hence False by simp  
 thus ?thesis ..

next

assume "(the (nhop (rt ( $\sigma i$ )) dip))  $\neq$  i"  
 with  $\langle \forall j. j \neq i \rightarrow \sigma j = \sigma' j \rangle$   
 have \*: " $\sigma$  (the (nhop (rt ( $\sigma i$ )) dip)) =  $\sigma'$  (the (nhop (rt ( $\sigma i$ )) dip))" by simp  
 with  $\langle \text{dip} \in \text{vD}(\text{rt}(\sigma' (\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))) \rangle$   
 have "dip  $\in$  vD(rt( $\sigma$  (the (nhop (rt ( $\sigma i$ )) dip))))" by simp  
 with a1 a2 a4 have "rt( $\sigma i$ )  $\sqsubseteq_{\text{dip}}$  rt( $\sigma$  (the (nhop (rt ( $\sigma i$ )) dip)))" by simp  
 with \* show ?thesis by simp

qed

qed

} note basic = this

{ fix  $\sigma \sigma'$  a dip sip i

assume a1: " $\forall \text{dip}. \text{dip} \in \text{vD}(\text{rt}(\sigma i))$   
 $\wedge \text{dip} \in \text{vD}(\text{rt}(\sigma (\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$   
 $\wedge \text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}$   
 $\rightarrow \text{rt}(\sigma i) \sqsubseteq_{\text{dip}} \text{rt}(\sigma (\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))"$

and ow: "?S i  $\sigma \sigma'$  a"

have " $\forall \text{dip}. \text{dip} \in \text{vD}(\text{update}(\text{rt}(\sigma i)) \text{sip} (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \}))$   
 $\wedge \text{dip} \in \text{vD}(\text{rt}(\sigma' (\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{sip} (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \})) \text{dip}))))$   
 $\wedge \text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{sip} (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \})) \text{dip}) \neq \text{dip}$   
 $\rightarrow \text{update}(\text{rt}(\sigma i)) \text{sip} (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \})$   
 $\sqsubseteq_{\text{dip}} \text{rt}(\sigma' (\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{sip} (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \})) \text{dip})))"$

proof clarify

fix dip

assume a2: "dip  $\in$  vD(update (rt ( $\sigma i$ )) sip (0, unk, val, Suc 0, sip, { }))"  
 and a3: "dip  $\in$  vD(rt ( $\sigma'$  (the (nhop  
 (update (rt ( $\sigma i$ )) sip (0, unk, val, Suc 0, sip, { })) dip))))"  
 and a4: "the (nhop (update (rt ( $\sigma i$ )) sip (0, unk, val, Suc 0, sip, { })) dip)  $\neq$  dip"  
 show "update (rt ( $\sigma i$ )) sip (0, unk, val, Suc 0, sip, { })"

```

     $\sqsubset_{dip}$  rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, { }))) dip)))"
proof (cases "dip = sip")
  assume "dip = sip"
  with (the (nhop (update (rt ( $\sigma$  i)) sip (0, unk, val, Suc 0, sip, { }))) dip)  $\neq$  dip)
  have False by simp
  thus ?thesis ..
next
  assume [simp]: "dip  $\neq$  sip"
  from a2 have "dip  $\in$  vD(rt ( $\sigma$  i))  $\vee$  dip = sip"
  by (rule vD_update_val)
  with (dip  $\neq$  sip) have "dip  $\in$  vD(rt ( $\sigma$  i))" by simp
  moreover from a3 have "dip  $\in$  vD(rt ( $\sigma'$  (the (nhop (rt ( $\sigma$  i)) dip))))" by simp
  moreover from a4 have "the (nhop (rt ( $\sigma$  i)) dip)  $\neq$  dip" by simp
  ultimately have "rt ( $\sigma$  i)  $\sqsubset_{dip}$  rt ( $\sigma'$  (the (nhop (rt ( $\sigma$  i)) dip)))"
  using a1 ow by - (drule(1) basic, simp)
  with (dip  $\neq$  sip) show ?thesis
  by - (erule rt_strictly_fresher_update_other, simp)
qed
qed
} note update_0_unk = this

{ fix  $\sigma$  a  $\sigma'$  nhop
  assume pre: " $\forall$ dip. dip  $\in$  vD(rt ( $\sigma$  i))  $\wedge$  dip  $\in$  vD(rt ( $\sigma$  (nhop dip)))  $\wedge$  nhop dip  $\neq$  dip
     $\longrightarrow$  rt ( $\sigma$  i)  $\sqsubset_{dip}$  rt ( $\sigma$  (nhop dip))"
  and ow: "?S i  $\sigma$   $\sigma'$  a"
  have " $\forall$ dip. dip  $\in$  vD (invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i)))
     $\wedge$  dip  $\in$  vD (rt ( $\sigma'$  (nhop dip)))  $\wedge$  nhop dip  $\neq$  dip
     $\longrightarrow$  rt ( $\sigma$  i)  $\sqsubset_{dip}$  rt ( $\sigma'$  (nhop dip))"
proof clarify
  fix dip
  assume "dip  $\in$  vD(invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i)))"
  and "dip  $\in$  vD(rt ( $\sigma'$  (nhop dip)))"
  and "nhop dip  $\neq$  dip"
  from this(1) have "dip  $\in$  vD (rt ( $\sigma$  i))"
  by (clarsimp dest!: vD_invalidate_vD_not_dests)
  moreover from ow have " $\forall$ j. j  $\neq$  i  $\longrightarrow$   $\sigma$  j =  $\sigma'$  j" by auto
  ultimately have "rt ( $\sigma$  i)  $\sqsubset_{dip}$  rt ( $\sigma$  (nhop dip))"
  using pre (dip  $\in$  vD (rt ( $\sigma'$  (nhop dip)))) (nhop dip  $\neq$  dip)
  by metis
  with ( $\forall$ j. j  $\neq$  i  $\longrightarrow$   $\sigma$  j =  $\sigma'$  j) show "rt ( $\sigma$  i)  $\sqsubset_{dip}$  rt ( $\sigma'$  (nhop dip))"
  by (metis rt_strictly_fresher_irefl)
qed
} note invalidate = this

{ fix  $\sigma$  a  $\sigma'$  dip oip osn sip hops i
  assume pre: " $\forall$ dip. dip  $\in$  vD (rt ( $\sigma$  i))
     $\wedge$  dip  $\in$  vD (rt ( $\sigma$  (the (nhop (rt ( $\sigma$  i)) dip))))
     $\wedge$  the (nhop (rt ( $\sigma$  i)) dip)  $\neq$  dip
     $\longrightarrow$  rt ( $\sigma$  i)  $\sqsubset_{dip}$  rt ( $\sigma$  (the (nhop (rt ( $\sigma$  i)) dip))))"
  and ow: "?S i  $\sigma$   $\sigma'$  a"
  and "Suc 0  $\leq$  osn"
  and a6: "sip  $\neq$  oip  $\longrightarrow$  oip  $\in$  kD (rt ( $\sigma$  sip))
     $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma$  sip)) oip
     $\wedge$  (nsqn (rt ( $\sigma$  sip)) oip = osn
     $\longrightarrow$  the (dhops (rt ( $\sigma$  sip)) oip)  $\leq$  hops
     $\vee$  the (flag (rt ( $\sigma$  sip)) oip) = inv)"
  and after: " $\sigma'$  i =  $\sigma$  i (|rt := update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { })|)"
  have " $\forall$ dip. dip  $\in$  vD (update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { })))
     $\wedge$  dip  $\in$  vD (rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) oip
      (osn, kno, val, Suc hops, sip, { }))) dip))))
     $\wedge$  the (nhop (update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { }))) dip)  $\neq$  dip
     $\longrightarrow$  update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { })
     $\sqsubset_{dip}$ 
    rt ( $\sigma'$  (the (nhop (update (rt ( $\sigma$  i)) oip (osn, kno, val, Suc hops, sip, { }))) dip)))"

```

```

proof clarify
  fix dip
  assume a2: "dip∈vD(update (rt (σ i)) oip (osn, kno, val, Suc (hops), sip, { })))"
    and a3: "dip∈vD(rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip, { }))) dip))))"
    and a4: "the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }))) dip) ≠ dip"
  from ow have a5: "∀j. j ≠ i → σ j = σ' j" by auto
  show "update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })
    □dip
    rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }))) dip)))"
    (is "?rt1 □dip ?rt2 dip")
  proof (cases "?rt1 = rt (σ i)")
    assume nochange [simp]:
      "update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }) = rt (σ i)"

  from after have "σ' i = σ i" by simp
  with a5 have "∀j. σ j = σ' j" by metis

  from a2 have "dip∈vD (rt (σ i))" by simp
  moreover from a3 have "dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))"
    using nochange and ⟨∀j. σ j = σ' j⟩ by clarsimp
  moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
  ultimately have "rt (σ i) □dip rt (σ (the (nhop (rt (σ i)) dip)))"
    using pre by simp

  hence "rt (σ i) □dip rt (σ' (the (nhop (rt (σ i)) dip)))"
    using ⟨∀j. σ j = σ' j⟩ by simp
  thus "?thesis" by simp
next
  assume change: "?rt1 ≠ rt (σ i)"
  from after a2 have "dip∈kD(rt (σ' i))" by auto
  show ?thesis
  proof (cases "dip = oip")
    assume "dip ≠ oip"

    with a2 have "dip∈vD (rt (σ i))" by auto
    moreover with a3 a5 after and ⟨dip ≠ oip⟩
      have "dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))"
        by simp metis
    moreover from a4 and ⟨dip ≠ oip⟩ have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
    ultimately have "rt (σ i) □dip rt (σ (the (nhop (rt (σ i)) dip)))"
      using pre by simp

    with after and a5 and ⟨dip ≠ oip⟩ show ?thesis
      by simp (metis rt_strictly_fresher_update_other
        rt_strictly_fresher_irefl)
  next
    assume "dip = oip"

    with a4 and change have "sip ≠ oip" by simp
    with a6 have "oip∈kD(rt (σ sip))"
      and "osn ≤ nsqn (rt (σ sip)) oip" by auto

    from a3 change ⟨dip = oip⟩ have "oip∈vD(rt (σ' sip))" by simp
    hence "the (flag (rt (σ' sip)) oip) = val" by simp

    from ⟨oip∈kD(rt (σ sip))⟩
      have "osn < nsqn (rt (σ' sip)) oip ∨ (osn = nsqn (rt (σ' sip)) oip
        ∧ the (dhops (rt (σ' sip)) oip) ≤ hops)"
    proof
      assume "oip∈vD(rt (σ sip))"
      hence "the (flag (rt (σ sip)) oip) = val" by simp
      with a6 ⟨sip ≠ oip⟩ have "nsqn (rt (σ sip)) oip = osn →
        the (dhops (rt (σ sip)) oip) ≤ hops"
    end
  end

```

```

    by simp
show ?thesis
proof (cases "sip = i")
  assume "sip ≠ i"
  with a5 have "σ sip = σ' sip" by simp
  with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
    and ⟨nsqn (rt (σ sip)) oip = osn ⟶ the (dhops (rt (σ sip)) oip) ≤ hops⟩
  show ?thesis by auto
next
— alternative to using sip_not_ip
assume [simp]: "sip = i"
have "?rt1 = rt (σ i)"
proof (rule update_cases_kD, simp_all)
  from ⟨Suc 0 ≤ osn⟩ show "0 < osn" by simp
next
  from ⟨oip ∈ kD (rt (σ sip))⟩ and ⟨sip = i⟩ show "oip ∈ kD (rt (σ i))"
  by simp
next
  assume "sqn (rt (σ i)) oip < osn"
  also from ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
    have "... ≤ nsqn (rt (σ i)) oip" by simp
  also have "... ≤ sqn (rt (σ i)) oip"
    by (rule nsqn_sqn)
  finally have "sqn (rt (σ i)) oip < sqn (rt (σ i)) oip" .
  hence False by simp
  thus "(λa. if a = oip
    then Some (osn, kno, val, Suc hops, i, π₇ (the (rt (σ i) oip)))
    else rt (σ i) a) = rt (σ i)" ..
next
  assume "sqn (rt (σ i)) oip = osn"
    and "Suc hops < the (dhops (rt (σ i)) oip)"
  from this(1) and ⟨oip ∈ vD (rt (σ sip))⟩ have "nsqn (rt (σ i)) oip = osn"
  by simp
  with ⟨nsqn (rt (σ sip)) oip = osn ⟶ the (dhops (rt (σ sip)) oip) ≤ hops⟩
    have "the (dhops (rt (σ i)) oip) ≤ hops" by simp
  with ⟨Suc hops < the (dhops (rt (σ i)) oip)⟩ have False by simp
  thus "(λa. if a = oip
    then Some (osn, kno, val, Suc hops, i, π₇ (the (rt (σ i) oip)))
    else rt (σ i) a) = rt (σ i)" ..
next
  assume "the (flag (rt (σ i)) oip) = inv"
  with ⟨the (flag (rt (σ sip)) oip) = val⟩ have False by simp
  thus "(λa. if a = oip
    then Some (osn, kno, val, Suc hops, i, π₇ (the (rt (σ i) oip)))
    else rt (σ i) a) = rt (σ i)" ..
next
  from ⟨oip ∈ kD (rt (σ sip))⟩
    show "(λa. if a = oip then Some (the (rt (σ i) oip)) else rt (σ i) a) = rt (σ i)"
    by (auto dest!: kD_Some)
qed
with change have False ..
thus ?thesis ..
qed
next
assume "oip ∈ iD (rt (σ sip))"
with ⟨the (flag (rt (σ' sip)) oip) = val⟩ and a5 have "sip = i"
  by (metis f.distinct(1) iD_flag_is_inv)
from ⟨oip ∈ iD (rt (σ sip))⟩ have "the (flag (rt (σ sip)) oip) = inv" by auto
with ⟨sip = i⟩ ⟨Suc 0 ≤ osn⟩ change after ⟨oip ∈ kD (rt (σ sip))⟩
  have "nsqn (rt (σ sip)) oip < nsqn (rt (σ' sip)) oip"
    unfolding update_def
  by (clarsimp split: option.split_asm split_if_asm)
  (auto simp: sqn_def)
with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩ have "osn < nsqn (rt (σ' sip)) oip"

```

```

    by simp
  thus ?thesis ..
qed
thus ?thesis
proof
  assume osnlt: "osn < nsqn (rt (σ' sip)) oip"
  from ⟨dip ∈ kD(rt (σ' i))⟩ and ⟨dip = oip⟩ have "dip ∈ kD (?rt1)" by simp
  moreover from a3 have "dip ∈ kD(?rt2 dip)" by simp
  moreover have "nsqn ?rt1 dip < nsqn (?rt2 dip) dip"
  proof -
    have "nsqn ?rt1 oip = osn"
    by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
    also have "... < nsqn (rt (σ' sip)) oip" using osnlt .
    also have "... = nsqn (?rt2 oip) oip" by (simp add: change)
    finally show ?thesis
    using ⟨dip = oip⟩ by simp
  qed
  ultimately show ?thesis
  by (rule rt_strictly_fresher_ltI)
next
  assume osneq: "osn = nsqn (rt (σ' sip)) oip ∧ the (dhops (rt (σ' sip)) oip) ≤ hops"

  have "oip ∈ kD(?rt1)" by simp
  moreover from a3 ⟨dip = oip⟩ have "oip ∈ kD(?rt2 oip)" by simp

  moreover have "nsqn ?rt1 oip = nsqn (?rt2 oip) oip"
  proof -
    from osneq have "osn = nsqn (rt (σ' sip)) oip" ..
    also have "osn = nsqn ?rt1 oip"
    by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
    also have "nsqn (rt (σ' sip)) oip = nsqn (?rt2 oip) oip"
    by (simp add: change)
    finally show ?thesis .
  qed

  moreover have "π5(the (?rt2 oip oip)) < π5(the (?rt1 oip))"
  proof -
    from osneq have "the (dhops (rt (σ' sip)) oip) ≤ hops" ..
    moreover from ⟨oip ∈ vD (rt (σ' sip))⟩ have "oip ∈ kD(rt (σ' sip))" by auto
    ultimately have "π5(the (rt (σ' sip) oip)) ≤ hops"
    by (auto simp add: proj5_eq_dhops)
    also from change after have "hops < π5(the (rt (σ' i) oip))"
    by (simp add: proj5_eq_dhops) (metis dhops_update_changed lessI)
    finally have "π5(the (rt (σ' sip) oip)) < π5(the (rt (σ' i) oip))" .
    with change after show ?thesis by simp
  qed

  ultimately have "?rt1 ⊆oip ?rt2 oip"
  by (rule rt_strictly_fresher_eqI)
  with ⟨dip = oip⟩ show ?thesis by simp
  qed
qed
qed
qed
} note rreq_rrep_update = this

have "opaadv i ⊨ (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
                                                ∧ msg_zhops m))),
      other quality_increases {i} →)
  onl ΓAODV
  (λ(σ, _). ∀dip. dip ∈ vD (rt (σ i)) ∩ vD (rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip))))"
proof (inv_cterms inv add: onl_oInvariant_sterms [OF adv_wf rreq_sip [THEN weaken]])

```

```

      onl_oinvariant_sterms [OF aadv_wf rrep_sip [THEN weaken]]
      onl_oinvariant_sterms [OF aadv_wf rerr_sip [THEN weaken]]
      onl_oinvariant_sterms [OF aadv_wf oosn_rreq [THEN weaken]]
      onl_oinvariant_sterms [OF aadv_wf odsn_rrep [THEN weaken]]
      onl_oinvariant_sterms [OF aadv_wf oadpreRT_welldefined]
    solve: basic update_0_unk invalidate rreq_rrep_update
    simp add: seqlsimp)
fix  $\sigma \sigma' p l$ 
assume or: " $(\sigma, p) \in \text{oreachable} (\text{opaadv } i) (?S \ i) \ (\text{other quality\_increases } \{i\})$ "
  and "other quality\_increases  $\{i\} \ \sigma \ \sigma'$ "
  and ll: " $l \in \text{labels } \Gamma_{AODV} \ p$ "
  and pre: " $\forall \text{dip. dip} \in vD \ (\text{rt } (\sigma \ i))$ 
     $\wedge \text{dip} \in vD(\text{rt } (\sigma \ (\text{the } (\text{nhop } (\text{rt } (\sigma \ i)) \ \text{dip}))))$ 
     $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma \ i)) \ \text{dip}) \neq \text{dip}$ 
     $\longrightarrow \text{rt } (\sigma \ i) \sqsubset_{\text{dip}} \text{rt } (\sigma \ (\text{the } (\text{nhop } (\text{rt } (\sigma \ i)) \ \text{dip})))$ "
from this(1-2)
  have or': " $(\sigma', p) \in \text{oreachable} (\text{opaadv } i) (?S \ i) \ (\text{other quality\_increases } \{i\})$ "
    by - (rule oreachable_other')

from or and ll have next_hop: " $\forall \text{dip. let } \text{nhip} = \text{the } (\text{nhop } (\text{rt } (\sigma \ i)) \ \text{dip})$ 
  in  $\text{dip} \in kD(\text{rt } (\sigma \ i)) \wedge \text{nhip} \neq \text{dip}$ 
   $\longrightarrow \text{dip} \in kD(\text{rt } (\sigma \ \text{nhip}))$ 
   $\wedge \text{nsqn } (\text{rt } (\sigma \ i)) \ \text{dip} \leq \text{nsqn } (\text{rt } (\sigma \ \text{nhip})) \ \text{dip}$ "
  by (auto dest!: onl_oinvariant_weakenD [OF seq_compare_next_hop'])

from or and ll have unk_hops_one: " $\forall \text{dip} \in kD \ (\text{rt } (\sigma \ i)). \ \text{sqn } (\text{rt } (\sigma \ i)) \ \text{dip} = 0$ 
   $\longrightarrow \text{sqnf } (\text{rt } (\sigma \ i)) \ \text{dip} = \text{unk}$ 
   $\wedge \text{the } (\text{dhops } (\text{rt } (\sigma \ i)) \ \text{dip}) = 1$ 
   $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma \ i)) \ \text{dip}) = \text{dip}$ "
  by (auto dest!: onl_oinvariant_weakenD [OF ozero_seq_unk_hops_one
    [OF oaadv_trans aadv_trans]]
    otherwith_actionD
    simp: seqlsimp)

from  $\langle \text{other quality\_increases } \{i\} \ \sigma \ \sigma' \rangle$  have " $\sigma' \ i = \sigma \ i$ " by auto
hence "quality\_increases  $(\sigma \ i) \ (\sigma' \ i)$ " by auto
with  $\langle \text{other quality\_increases } \{i\} \ \sigma \ \sigma' \rangle$  have " $\forall j. \ \text{quality\_increases } (\sigma \ j) \ (\sigma' \ j)$ "
  by - (erule otherE, metis singleton_iff)

show " $\forall \text{dip. dip} \in vD \ (\text{rt } (\sigma' \ i))$ 
   $\wedge \text{dip} \in vD \ (\text{rt } (\sigma' \ (\text{the } (\text{nhop } (\text{rt } (\sigma' \ i)) \ \text{dip}))))$ 
   $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma' \ i)) \ \text{dip}) \neq \text{dip}$ 
   $\longrightarrow \text{rt } (\sigma' \ i) \sqsubset_{\text{dip}} \text{rt } (\sigma' \ (\text{the } (\text{nhop } (\text{rt } (\sigma' \ i)) \ \text{dip})))$ "
proof clarify
  fix dip
  assume "dip  $\in vD(\text{rt } (\sigma' \ i))$ "
  and "dip  $\in vD(\text{rt } (\sigma' \ (\text{the } (\text{nhop } (\text{rt } (\sigma' \ i)) \ \text{dip}))))$ "
  and "the  $(\text{nhop } (\text{rt } (\sigma' \ i)) \ \text{dip}) \neq \text{dip}$ "
  from this(1) and  $\langle \sigma' \ i = \sigma \ i \rangle$  have "dip  $\in vD(\text{rt } (\sigma \ i))$ "
    and "dip  $\in kD(\text{rt } (\sigma \ i))$ "
  by auto

from  $\langle \text{the } (\text{nhop } (\text{rt } (\sigma' \ i)) \ \text{dip}) \neq \text{dip} \rangle$  and  $\langle \sigma' \ i = \sigma \ i \rangle$ 
  have "the  $(\text{nhop } (\text{rt } (\sigma \ i)) \ \text{dip}) \neq \text{dip}$ " (is "?nhop  $\neq$  _") by simp
with  $\langle \text{dip} \in kD(\text{rt } (\sigma \ i)) \rangle$  and next_hop
  have "dip  $\in kD(\text{rt } (\sigma \ (?nhip)))$ "
  and nsqns: "nsqn  $(\text{rt } (\sigma \ i)) \ \text{dip} \leq \text{nsqn } (\text{rt } (\sigma \ ?nhip)) \ \text{dip}$ "
  by (auto simp: Let_def)

have " $0 < \text{sqn } (\text{rt } (\sigma \ i)) \ \text{dip}$ "
  proof (rule neq0_conv [THEN iffD1, OF notI])
    assume "sqn  $(\text{rt } (\sigma \ i)) \ \text{dip} = 0$ "
    with  $\langle \text{dip} \in kD(\text{rt } (\sigma \ i)) \rangle$  and unk_hops_one
      have "?nhop = dip" by simp

```

```

    with ⟨?nhip ≠ dip⟩ show False ..
  qed
  also have "... = nsqn (rt (σ i)) dip"
    by (rule vD_nsqn_sqn [OF ⟨dip∈vD(rt (σ i))⟩, THEN sym])
  also have "... ≤ nsqn (rt (σ ?nhip)) dip"
    by (rule nsqns)
  also have "... ≤ sqn (rt (σ ?nhip)) dip"
    by (rule nsqn_sqn)
  finally have "0 < sqn (rt (σ ?nhip)) dip" .

  have "rt (σ i) ⊑dip rt (σ' ?nhip)"
  proof (cases "dip∈vD(rt (σ ?nhip))")
    assume "dip∈vD(rt (σ ?nhip))"
    with pre ⟨dip∈vD(rt (σ i))⟩ and ⟨?nhip ≠ dip⟩
      have "rt (σ i) ⊑dip rt (σ ?nhip)" by auto
    moreover from ⟨∀j. quality_increases (σ j) (σ' j)⟩
      have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
    ultimately show ?thesis
      using ⟨dip∈kD(rt (σ ?nhip))⟩
      by (rule strictly_fresher_quality_increases_right)
  next
    assume "dip∉vD(rt (σ ?nhip))"
    with ⟨dip∈kD(rt (σ ?nhip))⟩ have "dip∈iD(rt (σ ?nhip))" ..
    hence "the (flag (rt (σ ?nhip)) dip) = inv"
      by auto
    have "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ ?nhip)) dip"
      by (rule nsqns)
    also from ⟨dip∈iD(rt (σ ?nhip))⟩
      have "... = sqn (rt (σ ?nhip)) dip - 1" ..
    also have "... < sqn (rt (σ' ?nhip)) dip"
      proof -
        from ⟨∀j. quality_increases (σ j) (σ' j)⟩
          have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
        hence "∀ip. sqn (rt (σ ?nhip)) ip ≤ sqn (rt (σ' ?nhip)) ip" by auto
        hence "sqn (rt (σ ?nhip)) dip ≤ sqn (rt (σ' ?nhip)) dip" ..
        with ⟨0 < sqn (rt (σ ?nhip)) dip⟩ show ?thesis by auto
      qed
    also have "... = nsqn (rt (σ' ?nhip)) dip"
      proof (rule vD_nsqn_sqn [THEN sym])
        from ⟨dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and ⟨σ' i = σ i⟩
          show "dip∈vD(rt (σ' ?nhip))" by simp
      qed
    finally have "nsqn (rt (σ i)) dip < nsqn (rt (σ' ?nhip)) dip" .

    moreover from ⟨dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and ⟨σ' i = σ i⟩
      have "dip∈kD(rt (σ' ?nhip))" by auto
    ultimately show "rt (σ i) ⊑dip rt (σ' ?nhip)"
      using ⟨dip∈kD(rt (σ i))⟩ by - (rule rt_strictly_fresher_ltI)
  qed
  with ⟨σ' i = σ i⟩ show "rt (σ' i) ⊑dip rt (σ' (the (nhop (rt (σ' i)) dip)))"
    by simp
  qed
  qed
  thus ?thesis unfolding Let_def .
  qed

```

lemma seq\_compare\_next\_hop:

fixes w

shows "opaadv i ⊨ (otherwith (op=) {i} (orecvmsg msg\_fresh),  
 other quality\_increases {i} →)  
 global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)  
 in dip ∈ kD(rt (σ i)) ∧ nhip ≠ dip →  
 dip ∈ kD(rt (σ nhip))  
 ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip)"

by (rule oinvariant\_weakenE [OF seq\_compare\_next\_hop']) (auto dest!: onID)

lemma seq\_nhop\_quality\_increases:

shows "opaadv i  $\models$  (otherwith (op=) {i}  
 (orecvmsg ( $\lambda\sigma m$ . msg\_fresh  $\sigma m \wedge$  msg\_zhops m)),  
 other quality\_increases {i}  $\rightarrow$ )  
 global ( $\lambda\sigma$ .  $\forall$ dip. let nhip = the (nhop (rt ( $\sigma$  i)) dip)  
 in dip  $\in$  vD (rt ( $\sigma$  i))  $\cap$  vD (rt ( $\sigma$  nhip))  $\wedge$  nhip  $\neq$  dip  
 $\rightarrow$  (rt ( $\sigma$  i))  $\sqsubset_{\text{dip}}$  (rt ( $\sigma$  nhip)))")

by (rule oinvariant\_weakenE [OF seq\_nhop\_quality\_increases']) (auto dest!: onID)

end

## 2.10 Routing graphs and loop freedom

theory B\_Loop\_Freedom

imports B\_Aodv\_Predicates B\_Fresher

begin

Define the central theorem that relates an invariant over network states to the absence of loops in the associate routing graph.

definition

rt\_graph :: "(ip  $\Rightarrow$  state)  $\Rightarrow$  ip  $\Rightarrow$  ip rel"

where

"rt\_graph  $\sigma$  = ( $\lambda$ dip.  
 {(ip, ip') | ip ip' dsn dsk hops pre.  
 ip  $\neq$  dip  $\wedge$  rt ( $\sigma$  ip) dip = Some (dsn, dsk, val, hops, ip', pre)})"

Given the state of a network  $\sigma$ , a routing graph for a given destination ip address  $dip$  abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

lemma rt\_graphE [elim]:

fixes n dip ip ip'  
 assumes "(ip, ip')  $\in$  rt\_graph  $\sigma$  dip"  
 shows "ip  $\neq$  dip  $\wedge$  ( $\exists$ r. rt ( $\sigma$  ip) = r  
 $\wedge$  ( $\exists$ dsn dsk hops pre. r dip = Some (dsn, dsk, val, hops, ip', pre)))"  
 using assms unfolding rt\_graph\_def by auto

lemma rt\_graph\_vD [dest]:

" $\bigwedge$ ip ip'  $\sigma$  dip. (ip, ip')  $\in$  rt\_graph  $\sigma$  dip  $\implies$  dip  $\in$  vD(rt ( $\sigma$  ip))"  
 unfolding rt\_graph\_def vD\_def by auto

lemma rt\_graph\_vD\_trans [dest]:

" $\bigwedge$ ip ip'  $\sigma$  dip. (ip, ip')  $\in$  (rt\_graph  $\sigma$  dip)<sup>+</sup>  $\implies$  dip  $\in$  vD(rt ( $\sigma$  ip))"  
 by (erule converse\_tranclE) auto

lemma rt\_graph\_not\_dip [dest]:

" $\bigwedge$ ip ip'  $\sigma$  dip. (ip, ip')  $\in$  rt\_graph  $\sigma$  dip  $\implies$  ip  $\neq$  dip"  
 unfolding rt\_graph\_def by auto

lemma rt\_graph\_not\_dip\_trans [dest]:

" $\bigwedge$ ip ip'  $\sigma$  dip. (ip, ip')  $\in$  (rt\_graph  $\sigma$  dip)<sup>+</sup>  $\implies$  ip  $\neq$  dip"  
 by (erule converse\_tranclE) auto

NB: the property below cannot be lifted to the transitive closure

lemma rt\_graph\_nhip\_is\_nhop [dest]:

" $\bigwedge$ ip ip'  $\sigma$  dip. (ip, ip')  $\in$  rt\_graph  $\sigma$  dip  $\implies$  ip' = the (nhop (rt ( $\sigma$  ip)) dip)"  
 unfolding rt\_graph\_def by auto

theorem inv\_to\_loop\_freedom:

assumes " $\forall$ i dip. let nhip = the (nhop (rt ( $\sigma$  i)) dip)  
 in dip  $\in$  vD (rt ( $\sigma$  i))  $\cap$  vD (rt ( $\sigma$  nhip))  $\wedge$  nhip  $\neq$  dip  
 $\rightarrow$  (rt ( $\sigma$  i))  $\sqsubset_{\text{dip}}$  (rt ( $\sigma$  nhip))"  
 shows " $\forall$ dip. irrefl ((rt\_graph  $\sigma$  dip)<sup>+</sup>)"



```

using assms proof (intro allI)
  fix  $\sigma$  :: "ip  $\Rightarrow$  state" and dip
  assume inv: " $\forall$  ip dip.
    let nhip = the (nhop (rt ( $\sigma$  ip)) dip)
    in dip  $\in$  vD(rt ( $\sigma$  ip))  $\cap$  vD(rt ( $\sigma$  nhip))  $\wedge$ 
      nhip  $\neq$  dip  $\longrightarrow$  rt ( $\sigma$  ip)  $\sqsubset_{dip}$  rt ( $\sigma$  nhip)"
  { fix ip ip'
    assume "(ip, ip')  $\in$  (rt_graph  $\sigma$  dip)+"
      and "dip  $\in$  vD(rt ( $\sigma$  ip'))"
      and "ip'  $\neq$  dip"
    hence "rt ( $\sigma$  ip)  $\sqsubset_{dip}$  rt ( $\sigma$  ip'"
    proof induction
      fix nhip
      assume "(ip, nhip)  $\in$  rt_graph  $\sigma$  dip"
        and "dip  $\in$  vD(rt ( $\sigma$  nhip))"
        and "nhip  $\neq$  dip"
      from  $\langle$ (ip, nhip)  $\in$  rt_graph  $\sigma$  dip $\rangle$  have "dip  $\in$  vD(rt ( $\sigma$  ip))"
        and "nhip = the (nhop (rt ( $\sigma$  ip)) dip)"

        by auto
      from  $\langle$ dip  $\in$  vD(rt ( $\sigma$  ip)) $\rangle$  and  $\langle$ dip  $\in$  vD(rt ( $\sigma$  nhip)) $\rangle$ 
        have "dip  $\in$  vD(rt ( $\sigma$  ip))  $\cap$  vD(rt ( $\sigma$  nhip))" ..
      with  $\langle$ nhip = the (nhop (rt ( $\sigma$  ip)) dip) $\rangle$ 
        and  $\langle$ nhip  $\neq$  dip $\rangle$ 
        and inv
        show "rt ( $\sigma$  ip)  $\sqsubset_{dip}$  rt ( $\sigma$  nhip)"
        by (clarsimp simp: Let_def)
    next
      fix nhip nhip'
      assume "(ip, nhip)  $\in$  (rt_graph  $\sigma$  dip)+"
        and "(nhip, nhip')  $\in$  rt_graph  $\sigma$  dip"
        and IH: " $\llbracket$  dip  $\in$  vD(rt ( $\sigma$  nhip)); nhip  $\neq$  dip  $\rrbracket \Longrightarrow$  rt ( $\sigma$  ip)  $\sqsubset_{dip}$  rt ( $\sigma$  nhip)"
        and "dip  $\in$  vD(rt ( $\sigma$  nhip'))"
        and "nhip'  $\neq$  dip"
      from  $\langle$ (nhip, nhip')  $\in$  rt_graph  $\sigma$  dip $\rangle$  have 1: "dip  $\in$  vD(rt ( $\sigma$  nhip))"
        and 2: "nhip  $\neq$  dip"
        and "nhip' = the (nhop (rt ( $\sigma$  nhip)) dip)"

        by auto
      from 1 2 have "rt ( $\sigma$  ip)  $\sqsubset_{dip}$  rt ( $\sigma$  nhip)" by (rule IH)
      also have "rt ( $\sigma$  nhip)  $\sqsubset_{dip}$  rt ( $\sigma$  nhip'"
      proof -
        from  $\langle$ dip  $\in$  vD(rt ( $\sigma$  nhip)) $\rangle$  and  $\langle$ dip  $\in$  vD(rt ( $\sigma$  nhip')) $\rangle$ 
          have "dip  $\in$  vD(rt ( $\sigma$  nhip))  $\cap$  vD(rt ( $\sigma$  nhip'))" ..
        with  $\langle$ nhip'  $\neq$  dip $\rangle$ 
          and  $\langle$ nhip' = the (nhop (rt ( $\sigma$  nhip)) dip) $\rangle$ 
          and inv
          show "rt ( $\sigma$  nhip)  $\sqsubset_{dip}$  rt ( $\sigma$  nhip'"
          by (clarsimp simp: Let_def)
        qed
      finally show "rt ( $\sigma$  ip)  $\sqsubset_{dip}$  rt ( $\sigma$  nhip'" .
    qed } note fresher = this

show "irrefl ((rt_graph  $\sigma$  dip)+)"
unfolding irrefl_def proof (intro allI notI)
  fix ip
  assume "(ip, ip)  $\in$  (rt_graph  $\sigma$  dip)+"
  moreover then have "dip  $\in$  vD(rt ( $\sigma$  ip))"
    and "ip  $\neq$  dip"

  by auto
  ultimately have "rt ( $\sigma$  ip)  $\sqsubset_{dip}$  rt ( $\sigma$  ip)" by (rule fresher)
  thus False by simp
qed
qed
end

```

## 2.11 Lift and transfer invariants to show loop freedom

theory B\_Aodv\_Loop\_Freedom

imports "../.../AWN/OClosed\_Transfer" "../.../AWN/Qmsg\_Lifting" B\_Global\_Invariants B\_Loop\_Freedom  
begin

### 2.11.1 Lift to parallel processes with queues

lemma par\_step\_no\_change\_on\_send\_or\_receive:

fixes  $\sigma$   $s$   $a$   $\sigma'$   $s'$   
assumes " $((\sigma, s), a, (\sigma', s')) \in \text{oparp\_sos } i$  ( $\text{oseqp\_sos } \Gamma_{AODV} i$ ) ( $\text{seqp\_sos } \Gamma_{QMSG}$ )"  
and " $a \neq \tau$ "  
shows " $\sigma' i = \sigma i$ "  
using assms by (rule qmsg\_no\_change\_on\_send\_or\_receive)

lemma par\_nhop\_quality\_increases:

shows " $\text{opaodv } i \llbracket_i \text{qmsg} \models (\text{otherwith } (op=) \{i\} (\text{orecvmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)), \text{other quality\_increases } \{i\} \rightarrow) \text{global } (\lambda \sigma. \forall \text{dip. let } nhop = \text{the } (nhop \text{ (rt } (\sigma i)) \text{ dip}) \text{ in } \text{dip} \in \text{vD } (\text{rt } (\sigma i)) \cap \text{vD } (\text{rt } (\sigma nhop)) \wedge nhop \neq \text{dip} \rightarrow (\text{rt } (\sigma i)) \sqsubseteq_{\text{dip}} (\text{rt } (\sigma nhop)))$ "

proof (rule lift\_into\_qmsg [OF seq\_nhop\_quality\_increases])

show " $\text{opaodv } i \models_A (\text{otherwith } (op=) \{i\} (\text{orecvmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)), \text{other quality\_increases } \{i\} \rightarrow) \text{globala } (\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

proof (rule ostep\_invariant\_weakenE [OF oquality\_increases], simp\_all)

fix  $t$  :: " $((\text{nat} \Rightarrow \text{state}) \times (\text{state}, \text{msg}, \text{pseqp}, \text{pseqp label}) \text{seqp}), \text{msg seq\_action}) \text{transition}$ "  
assume " $\text{onll } \Gamma_{AODV} (\lambda((\sigma, \_), \_, (\sigma', \_)). \forall j. \text{quality\_increases } (\sigma j) (\sigma' j)) t$ "  
thus " $\text{quality\_increases } (\text{fst } (\text{fst } t) i) (\text{fst } (\text{snd } (\text{snd } t)) i)$ "  
by (cases  $t$ ) (clarsimp dest!: onllD, metis aodv\_ex\_label)

next

fix  $\sigma$   $\sigma'$   $a$   
assume " $\text{otherwith } (op=) \{i\} (\text{orecvmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)) \sigma \sigma' a$ "  
thus " $\text{otherwith quality\_increases } \{i\} (\text{orecvmsg } (\lambda \_. \text{rreq\_rrep\_sn})) \sigma \sigma' a$ "  
by - (erule weaken\_otherwith, auto)

qed

qed auto

lemma par\_rreq\_rrep\_sn\_quality\_increases:

" $\text{opaodv } i \llbracket_i \text{qmsg} \models_A (\lambda \sigma \_. \text{orecvmsg } (\lambda \_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda \_ \_. \text{True}) \{i\} \rightarrow) \text{globala } (\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

proof -

have " $\text{opaodv } i \models_A (\lambda \sigma \_. \text{orecvmsg } (\lambda \_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda \_ \_. \text{True}) \{i\} \rightarrow) \text{globala } (\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

by (rule ostep\_invariant\_weakenE [OF olocal\_quality\_increases])  
(auto dest!: onllD seqllD elim!: aodv\_ex\_labelE)

hence " $\text{opaodv } i \llbracket_i \text{qmsg} \models_A (\lambda \sigma \_. \text{orecvmsg } (\lambda \_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda \_ \_. \text{True}) \{i\} \rightarrow) \text{globala } (\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

by (rule lift\_step\_into\_qmsg\_statelessassm) simp\_all

thus ?thesis by rule auto

qed

lemma par\_rreq\_rrep\_nsqn\_fresh\_any\_step:

shows " $\text{opaodv } i \llbracket_i \text{qmsg} \models_A (\lambda \sigma \_. \text{orecvmsg } (\lambda \_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda \_ \_. \text{True}) \{i\} \rightarrow) \text{globala } (\lambda(\sigma, a, \sigma'). \text{anycast } (\text{msg\_fresh } \sigma) a)$ "

proof -

have " $\text{opaodv } i \models_A (\lambda \sigma \_. (\text{orecvmsg } (\lambda \_. \text{rreq\_rrep\_sn})) \sigma, \text{other } (\lambda \_ \_. \text{True}) \{i\} \rightarrow) \text{globala } (\lambda(\sigma, a, \sigma'). \text{anycast } (\text{msg\_fresh } \sigma) a)$ "

proof (rule ostep\_invariant\_weakenE [OF rreq\_rrep\_nsqn\_fresh\_any\_step\_invariant])

fix  $t$

```

assume "onll  $\Gamma_{AODV}$  ( $\lambda((\sigma, \_), a, \_)$ . anycast (msg_fresh  $\sigma$ ) a) t"
thus "globala ( $\lambda(\sigma, a, \sigma')$ . anycast (msg_fresh  $\sigma$ ) a) t"
  by (cases t) (clarsimp dest!: onllD, metis aadv_ex_label)
qed auto
hence "opaadv i  $\langle\langle_i$  qmsg  $\models_A$  ( $\lambda\sigma \_.$  orecvmsg ( $\lambda\_.$  rreq_rrep_sn))  $\sigma$ , other ( $\lambda\_ \_.$  True)  $\{i\}$   $\rightarrow$ 
  globala ( $\lambda(\sigma, a, \sigma')$ . anycast (msg_fresh  $\sigma$ ) a)"
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

```

lemma par\_anycast\_msg\_zhops:

```

shows "opaadv i  $\langle\langle_i$  qmsg  $\models_A$  ( $\lambda\sigma \_.$  orecvmsg ( $\lambda\_.$  rreq_rrep_sn))  $\sigma$ , other ( $\lambda\_ \_.$  True)  $\{i\}$   $\rightarrow$ 
  globala ( $\lambda(\_, a, \_)$ . anycast msg_zhops a)"

```

proof -

```

from anycast_msg_zhops initiali_aadv oaadv_trans aadv_trans

```

```

have "opaadv i  $\models_A$  (act TT, other ( $\lambda\_ \_.$  True)  $\{i\}$   $\rightarrow$ )
  seqll i (onll  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_)$ . anycast msg_zhops a))"

```

```

  by (rule open_seq_step_invariant)

```

```

hence "opaadv i  $\models_A$  ( $\lambda\sigma \_.$  orecvmsg ( $\lambda\_.$  rreq_rrep_sn))  $\sigma$ , other ( $\lambda\_ \_.$  True)  $\{i\}$   $\rightarrow$ 
  globala ( $\lambda(\_, a, \_)$ . anycast msg_zhops a)"

```

proof (rule ostep\_invariant\_weakenE)

```

fix t :: "((nat  $\Rightarrow$  state)  $\times$  (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"

```

```

assume "seqll i (onll  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_)$ . anycast msg_zhops a)) t"

```

```

thus "globala ( $\lambda(\_, a, \_)$ . anycast msg_zhops a) t"

```

```

  by (cases t) (clarsimp dest!: seqllD onllD, metis aadv_ex_label)

```

```

qed simp_all

```

```

hence "opaadv i  $\langle\langle_i$  qmsg  $\models_A$  ( $\lambda\sigma \_.$  orecvmsg ( $\lambda\_.$  rreq_rrep_sn))  $\sigma$ , other ( $\lambda\_ \_.$  True)  $\{i\}$   $\rightarrow$ 
  globala ( $\lambda(\_, a, \_)$ . anycast msg_zhops a)"

```

```

  by (rule lift_step_into_qmsg_statelessassm) simp_all

```

```

thus ?thesis by rule auto

```

qed

## 2.11.2 Lift to nodes

lemma node\_step\_no\_change\_on\_send\_or\_receive:

```

assumes "(( $\sigma$ , NodeS i P R), a, ( $\sigma'$ , NodeS i' P' R'))  $\in$  onode_sos

```

```

  (oparp_sos i (oseqp_sos  $\Gamma_{AODV}$  i) (seqp_sos  $\Gamma_{QMSG}$ ))"

```

```

  and "a  $\neq$   $\tau$ "

```

```

  shows " $\sigma' i = \sigma i$ "

```

```

using assms

```

```

by (cases a) (auto elim!: par_step_no_change_on_send_or_receive)

```

lemma node\_nhop\_quality\_increases:

```

shows "< i : opaadv i  $\langle\langle_i$  qmsg : R  $\rangle_o$   $\models$ 
  (otherwith (op=)  $\{i\}$ 

```

```

  (oarrivmsg ( $\lambda\sigma m.$  msg_fresh  $\sigma m \wedge$  msg_zhops m)),

```

```

  other quality_increases  $\{i\}$ 

```

```

 $\rightarrow$  global ( $\lambda\sigma. \forall$  dip. let nhop = the (nhop (rt ( $\sigma i$ )) dip)

```

```

  in dip  $\in$  vD (rt ( $\sigma i$ ))  $\cap$  vD (rt ( $\sigma$  nhop))  $\wedge$  nhop  $\neq$  dip

```

```

 $\rightarrow$  (rt ( $\sigma i$ ))  $\sqsubset_{\text{dip}}$  (rt ( $\sigma$  nhop)))"

```

```

by (rule node_lift [OF par_nhop_quality_increases]) auto

```

lemma node\_quality\_increases:

```

"< i : opaadv i  $\langle\langle_i$  qmsg : R  $\rangle_o$   $\models_A$  ( $\lambda\sigma \_.$  oarrivmsg ( $\lambda\_.$  rreq_rrep_sn))  $\sigma$ ,
  other ( $\lambda\_ \_.$  True)  $\{i\}$   $\rightarrow$ 

```

```

  globala ( $\lambda(\sigma, \_, \sigma')$ . quality_increases ( $\sigma i$ ) ( $\sigma' i$ ))"

```

```

by (rule node_lift_step_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp

```

lemma node\_rreq\_rrep\_nsqn\_fresh\_any\_step:

```

shows "< i : opaadv i  $\langle\langle_i$  qmsg : R  $\rangle_o$   $\models_A$ 

```

```

  ( $\lambda\sigma \_.$  oarrivmsg ( $\lambda\_.$  rreq_rrep_sn))  $\sigma$ , other ( $\lambda\_ \_.$  True)  $\{i\}$   $\rightarrow$ 

```

```

  globala ( $\lambda(\sigma, a, \sigma')$ . castmsg (msg_fresh  $\sigma$ ) a)"

```

```

by (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_nsqn_fresh_any_step])

```

lemma node\_anycast\_msg\_zhops:

shows " $\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R \rangle_o \models_A$   
 $(\lambda \sigma \_ . \text{oarrivemsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda \_ \_ . \text{True}) \{i\} \rightarrow)$   
 $\text{globala } (\lambda (\_, a, \_). \text{castmsg } \text{msg\_zhops } a)$ "  
 by (rule node\_lift\_anycast\_statelessassm [OF par\_anycast\_msg\_zhops])

lemma node\_silent\_change\_only:

shows " $\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R_i \rangle_o \models_A$   $(\lambda \sigma \_ . \text{oarrivemsg } (\lambda \_ \_ . \text{True}) \sigma,$   
 $\text{other } (\lambda \_ \_ . \text{True}) \{i\} \rightarrow)$   
 $\text{globala } (\lambda (\sigma, a, \sigma'). a \neq \tau \rightarrow \sigma' i = \sigma i)$ "

proof (rule ostep\_invariantI, simp (no\_asm), rule impI)

fix  $\sigma \zeta a \sigma' \zeta'$

assume or: " $(\sigma, \zeta) \in \text{oreachable } (\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R_i \rangle_o$   
 $(\lambda \sigma \_ . \text{oarrivemsg } (\lambda \_ \_ . \text{True}) \sigma)$   
 $(\text{other } (\lambda \_ \_ . \text{True}) \{i\}))$ "

and tr: " $((\sigma, \zeta), a, (\sigma', \zeta')) \in \text{trans } (\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R_i \rangle_o$ "

and " $a \neq \tau_n$ "

from or obtain p R where " $\zeta = \text{NodeS } i \text{ p } R$ "

by - (drule node\_net\_state, metis)

with tr have " $((\sigma, \text{NodeS } i \text{ p } R), a, (\sigma', \zeta'))$

$\in \text{onode\_sos } (\text{oparp\_sos } i (\text{trans } (\text{opaadv } i)) (\text{trans } \text{qmsg}))$ "

by simp

thus " $\sigma' i = \sigma i$ " using  $\langle a \neq \tau_n \rangle$

by (cases rule: onode\_sos.cases)

(auto elim: qmsg\_no\_change\_on\_send\_or\_receive)

qed

### 2.11.3 Lift to partial networks

lemma arrive\_rreq\_rrep\_nsqn\_fresh\_inc\_sn [simp]:

assumes " $\text{oarrivemsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge P \sigma m) \sigma m$ "

shows " $\text{oarrivemsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma m$ "

using assms by (cases m) auto

lemma opnet\_nhop\_quality\_increases:

shows " $\text{opnet } (\lambda i. \text{opaadv } i \langle \langle_i \text{ qmsg} \rangle p \models$   
 $(\text{otherwith } (\text{op}=) (\text{net\_tree\_ips } p)$   
 $(\text{oarrivemsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)),$   
 $\text{other quality\_increases } (\text{net\_tree\_ips } p) \rightarrow)$   
 $\text{global } (\lambda \sigma. \forall i \in \text{net\_tree\_ips } p. \forall \text{dip}.$   
 $\text{let nhip} = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{dip})$   
 $\text{in } \text{dip} \in \text{vD } (\text{rt } (\sigma i)) \cap \text{vD } (\text{rt } (\sigma \text{nhip})) \wedge \text{nhip} \neq \text{dip}$   
 $\rightarrow (\text{rt } (\sigma i)) \sqsubseteq_{\text{dip}} (\text{rt } (\sigma \text{nhip}))$ )"

proof (rule pnet\_lift [OF node\_nhop\_quality\_increases])

fix i R

have " $\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R \rangle_o \models_A$   $(\lambda \sigma \_ . \text{oarrivemsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma,$   
 $\text{other } (\lambda \_ \_ . \text{True}) \{i\} \rightarrow) \text{globala } (\lambda (\sigma, a, \sigma').$   
 $\text{castmsg } (\lambda m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) a)$ "

proof (rule ostep\_invariantI, simp (no\_asm))

fix  $\sigma s a \sigma' s'$

assume or: " $(\sigma, s) \in \text{oreachable } (\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R \rangle_o$   
 $(\lambda \sigma \_ . \text{oarrivemsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma)$   
 $(\text{other } (\lambda \_ \_ . \text{True}) \{i\}))$ "

and tr: " $((\sigma, s), a, (\sigma', s')) \in \text{trans } (\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R \rangle_o$ "

and am: " $\text{oarrivemsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma a$ "

from or tr am have " $\text{castmsg } (\text{msg\_fresh } \sigma) a$ "

by (auto dest!: ostep\_invariantD [OF node\_rreq\_rrep\_nsqn\_fresh\_any\_step])

moreover from or tr am have " $\text{castmsg } (\text{msg\_zhops}) a$ "

by (auto dest!: ostep\_invariantD [OF node\_anycast\_msg\_zhops])

ultimately show " $\text{castmsg } (\lambda m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) a$ "

by (case\_tac a) auto

qed

thus " $\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R \rangle_o \models_A$   
 $(\lambda \sigma \_ . \text{oarrivemsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) \sigma,$

```

      other quality_increases {i} → globala (λ(σ, a, _).
        castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
    by rule auto
  next
  fix i R
  show "⟨i : opaadv i ⟨⟨i qmsg : R⟩_o ⟩_A
    (λσ _. oarrivmsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
      other quality_increases {i} →) globala (λ(σ, a, σ').
        a ≠ τ ∧ (∀i d. a = i:deliver(d)) → σ i = σ' i)"
    by (rule ostep_invariant_weakenE [OF node_silent_change_only]) auto
  next
  fix i R
  show "⟨i : opaadv i ⟨⟨i qmsg : R⟩_o ⟩_A
    (λσ _. oarrivmsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
      other quality_increases {i} →) globala (λ(σ, a, σ').
        a = τ ∨ (∃d. a = i:deliver(d)) → quality_increases (σ i) (σ' i))"
    by (rule ostep_invariant_weakenE [OF node_quality_increases]) auto
  qed simp_all

```

#### 2.11.4 Lift to closed networks

```

lemma onet_nhop_quality_increases:
  shows "oclosed (opnet (λi. opaadv i ⟨⟨i qmsg⟩ p)
    = (λ_ _ . True, other quality_increases (net_tree_ips p) →)
      global (λσ. ∀i∈net_tree_ips p. ∀dip.
        let nhop = the (nhop (rt (σ i)) dip)
        in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhop)) ∧ nhop ≠ dip
          → (rt (σ i)) ⊆dip (rt (σ nhop)))"
  (is "_ = (_, ?U →) ?inv")
proof (rule inclosed_closed)
  from opnet_nhop_quality_increases
  show "opnet (λi. opaadv i ⟨⟨i qmsg⟩ p)
    = (otherwith (op=) (net_tree_ips p) inclosed, ?U →) ?inv"
proof (rule oinvariant_weakenE)
  fix σ σ' :: "ip ⇒ state" and a :: "msg node_action"
  assume "otherwith (op=) (net_tree_ips p) inclosed σ σ' a"
  thus "otherwith (op=) (net_tree_ips p)
    (oarrivmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)) σ σ' a"
proof (rule otherwithEI)
  fix σ :: "ip ⇒ state" and a :: "msg node_action"
  assume "inclosed σ a"
  thus "oarrivmsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ a"
  proof (cases a)
    fix ii ni ms
    assume "a = ii-ni:arrive(ms)"
    moreover with (inclosed σ a) obtain d di where "ms = newpkt(d, di)"
    by (cases ms) auto
    ultimately show ?thesis by simp
  qed simp_all
  qed
  qed
  qed

```

#### 2.11.5 Transfer into the standard model

```

interpretation aadv_openproc: openproc paadv opaadv id
  where "aadv_openproc.initmissing = initmissing"
proof -
  show "openproc paadv opaadv id"
  proof unfold_locales
    fix i :: ip
    have "{(σ, ζ). (σ i, ζ) ∈ σAADV i ∧ (∀j. j ≠ i → σ j ∈ fst ' σAADV j)} ⊆ σAADV"
      unfolding σAADV_def σAADV'_def
      proof (rule equalityD1)

```

```

    show "∧f p. {(σ, ζ). (σ i, ζ) ∈ {(f i, p)} ∧ (∀j. j ≠ i
      → σ j ∈ fst ' {(f j, p)}')} = {(f, p)}"
      by (rule set_eqI) auto
  qed
  thus "{ (σ, ζ) | σ ζ s. s ∈ init (paadv i)
    ∧ (σ i, ζ) = id s
    ∧ (∀j. j≠i → σ j ∈ (fst o id) ' init (paadv j)) } ⊆ init (opaadv i)"
    by simp
next
  show "∀j. init (paadv j) ≠ {}"
    unfolding σ_AODV_def by simp
next
  fix i s a s' σ σ'
  assume "σ i = fst (id s)"
    and "σ' i = fst (id s'"
    and "(s, a, s') ∈ trans (paadv i)"
  then obtain q q' where "s = (σ i, q)"
    and "s' = (σ' i, q'"
    and "((σ i, q), a, (σ' i, q')) ∈ trans (paadv i)"
    by (cases s, cases s') auto
  from this(3) have "((σ, q), a, (σ', q')) ∈ trans (opaadv i)"
    by simp (rule open_seqp_action [OF aadv_wf])

  with ⟨s = (σ i, q)⟩ and ⟨s' = (σ' i, q')⟩
  show "((σ, snd (id s)), a, (σ', snd (id s')))) ∈ trans (opaadv i)"
    by simp
  qed
  then interpret op: openproc paadv opaadv id .
  have [simp]: "∧i. (SOME x. x ∈ (fst o id) ' init (paadv i)) = aadv_init i"
    unfolding σ_AODV_def by simp
  hence "∧i. openproc.initmissing paadv id i = initmissing i"
    unfolding op.initmissing_def op.someinit_def initmissing_def
    by (auto split: option.split)
  thus "openproc.initmissing paadv id = initmissing" ..
qed

interpretation aadv_openproc_par_qmsg: openproc_parq paadv opaadv id qmsg
where "aadv_openproc_par_qmsg.netglobal = netglobal"
  and "aadv_openproc_par_qmsg.initmissing = initmissing"
proof -
  show "openproc_parq paadv opaadv id qmsg"
    by (unfold_locales) simp
  then interpret opq: openproc_parq paadv opaadv id qmsg .

  have im: "∧σ. openproc.initmissing (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) σ
    = initmissing σ"
    unfolding opq.initmissing_def opq.someinit_def initmissing_def
    unfolding σ_AODV_def σ_QMSG_def by (clarsimp cong: option.case_cong)
  thus "openproc.initmissing (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) = initmissing"
    by (rule ext)
  have "∧P σ. openproc.netglobal (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) P σ
    = netglobal P σ"
    unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def
    unfolding σ_AODV_def σ_QMSG_def
    by (clarsimp cong: option.case_cong
      simp del: One_nat_def
      simp add: fst_initmissing_netgmap_default_aadv_init_netlift
      [symmetric, unfolded initmissing_def])
  thus "openproc.netglobal (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) = netglobal"
    by auto
  qed

lemma net_nhop_quality_increases:
  assumes "wf_net_tree n"

```

```

shows "closed (pnet (λi. paadv i ⟨⟨ qmsg⟩ n) ⊨ netglobal
      (λσ. ∀i dip. let nhip = the (nhop (rt (σ i)) dip)
                  in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
                  → (rt (σ i)) ⊑dip (rt (σ nhip)))"
(is "_ ⊨ netglobal (λσ. ∀i. ?inv σ i)")
proof -
  from ⟨wf_net_tree n⟩
  have proto: "closed (pnet (λi. paadv i ⟨⟨ qmsg⟩ n) ⊨ netglobal (λσ. ∀i∈net_tree_ips n. ∀dip.
    let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊑dip (rt (σ nhip)))"
  by (rule aadv_openproc_par_qmsg.close_opnet [OF _ onet_nhop_quality_increases])
show ?thesis
unfolding invariant_def opnet_sos.opnet_tau1
proof (rule, simp only: aadv_openproc_par_qmsg.netglobalsimp
  fst_initmissing_netgmap_pair_fst, rule allI)
  fix σ i
  assume sr: "σ ∈ reachable (closed (pnet (λi. paadv i ⟨⟨ qmsg⟩ n) TT"
  hence "∀i∈net_tree_ips n. ?inv (fst (initmissing (netgmap fst σ))) i"
  by - (drule invariantD [OF proto],
    simp only: aadv_openproc_par_qmsg.netglobalsimp
    fst_initmissing_netgmap_pair_fst)
  thus "?inv (fst (initmissing (netgmap fst σ))) i"
  proof (cases "i∈net_tree_ips n")
    assume "i∉net_tree_ips n"
    from sr have "σ ∈ reachable (pnet (λi. paadv i ⟨⟨ qmsg⟩ n) TT" ..
    hence "net_ips σ = net_tree_ips n" ..
    with ⟨i∉net_tree_ips n⟩ have "i∉net_ips σ" by simp
    hence "(fst (initmissing (netgmap fst σ))) i = aadv_init i"
    by simp
    thus ?thesis by simp
  qed metis
qed
qed

```

### 2.11.6 Loop freedom of AODV

```

theorem aadv_loop_freedom:
  assumes "wf_net_tree n"
  shows "closed (pnet (λi. paadv i ⟨⟨ qmsg⟩ n) ⊨ netglobal (λσ. ∀dip. irrefl ((rt_graph σ dip)+))"
  using assms by (rule aadv_openproc_par_qmsg.netglobal_weakenE
    [OF net_nhop_quality_increases inv_to_loop_freedom])
end

```

## Chapter 3

# Variant C: From Groupcast to Broadcast

Explanation [4, §10.4]: A node maintains a set of ‘precursor nodes’ for each of its valid routes. If the link to a route’s next hop is lost, an error message is groupcast to the associated precursor nodes. The idea is to reduce the number of messages received and handled. However, precursor lists are incomplete. They are updated only when a RREP message is sent. This can lead to packet loss. A possible solution is to abandon precursors and to replace every groupcast by a broadcast. At first glance this strategy seems to need more bandwidth, but this is not the case. Sending error messages to a set of precursors is implemented at the link layer by broadcasting the message anyway; a node receiving such a message then checks the header to determine whether it is one of the intended recipients. Instead of analysing the header only, a node can just as well read the message and decide whether the information contained in the message is of use. To be more precise: an error message is useful for a node if the node has established a route to one of the nodes listed in the message, and the next hop to a listed node is the sender of the error message. In case a node finds useful information inside the message, it should update its routing table and distribute another error message.

### 3.1 Predicates and functions used in the AODV model

```
theory C_Aodv_Data
imports C_Gtobcast
begin
```

#### 3.1.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

```
definition inc :: "sqn  $\Rightarrow$  sqn"
  where "inc sn  $\equiv$  if sn = 0 then sn else sn + 1"
```

```
lemma less_than_inc [simp]: "x  $\leq$  inc x"
  unfolding inc_def by simp
```

```
lemma inc_minus_suc_0 [simp]:
  "inc x - Suc 0 = x"
  unfolding inc_def by simp
```

```
lemma inc_never_one' [simp, intro]: "inc x  $\neq$  Suc 0"
  unfolding inc_def by simp
```

```
lemma inc_never_one [simp, intro]: "inc x  $\neq$  1"
  by simp
```

#### 3.1.2 Modelling Routes

A route is a 5-tuple,  $(dsn, dsk, flag, hops, nhop)$  where  $dsn$  is the ‘destination sequence number’,  $dsk$  is the ‘destination-sequence-number status’,  $flag$  is the route status,  $hops$  is the number of hops to the destination, and  $nhop$  is the next hop toward the destination. In this variant, the set of ‘precursor nodes’ is not modelled.

```
type_synonym r = "sqn  $\times$  k  $\times$  f  $\times$  nat  $\times$  ip"
```



```

definition proj2 :: "r ⇒ sqn" ("π2")
  where "π2 ≡ λ(dsn, _, _, _). dsn"

definition proj3 :: "r ⇒ k" ("π3")
  where "π3 ≡ λ(_, dsk, _, _). dsk"

definition proj4 :: "r ⇒ f" ("π4")
  where "π4 ≡ λ(_, _, flag, _, _). flag"

definition proj5 :: "r ⇒ nat" ("π5")
  where "π5 ≡ λ(_, _, _, hops, _). hops"

definition proj6 :: "r ⇒ ip" ("π6")
  where "π6 ≡ λ(_, _, _, _, nhip). nhip"

lemma projs [simp]:
  "π2(dsn, dsk, flag, hops, nhip) = dsn"
  "π3(dsn, dsk, flag, hops, nhip) = dsk"
  "π4(dsn, dsk, flag, hops, nhip) = flag"
  "π5(dsn, dsk, flag, hops, nhip) = hops"
  "π6(dsn, dsk, flag, hops, nhip) = nhip"
  by (clarsimp simp: proj2_def proj3_def proj4_def
    proj5_def proj6_def)+

```

```

lemma proj3_pred [intro]: "[[ P kno; P unk ] ⇒ P (π3 x)"
  by (rule k.induct)

```

```

lemma proj4_pred [intro]: "[[ P val; P inv ] ⇒ P (π4 x)"
  by (rule f.induct)

```

```

lemma proj6_pair_snd [simp]:
  fixes dsn' r
  shows "π6 (dsn', snd (r)) = π6(r)"
  by (cases r) simp

```

### 3.1.3 Routing Tables

Routing tables map ip addresses to route entries.

```

type_synonym rt = "ip → r"

```

```

syntax
  "_Sigma_route" :: "rt ⇒ ip → r" ("σroute'(_, _)'")

```

```

translations
  "σroute(rt, dip)" => "rt dip"

```

```

definition sqn :: "rt ⇒ ip ⇒ sqn"
  where "sqn rt dip ≡ case σroute(rt, dip) of Some r ⇒ π2(r) | None ⇒ 0"

```

```

definition sqnf :: "rt ⇒ ip ⇒ k"
  where "sqnf rt dip ≡ case σroute(rt, dip) of Some r ⇒ π3(r) | None ⇒ unk"

```

```

abbreviation flag :: "rt ⇒ ip → f"
  where "flag rt dip ≡ map_option π4 (σroute(rt, dip))"

```

```

abbreviation dhops :: "rt ⇒ ip → nat"
  where "dhops rt dip ≡ map_option π5 (σroute(rt, dip))"

```

```

abbreviation nhop :: "rt ⇒ ip → ip"
  where "nhop rt dip ≡ map_option π6 (σroute(rt, dip))"

```

```

definition vD :: "rt ⇒ ip set"
  where "vD rt ≡ {dip. flag rt dip = Some val}"

```

```

definition iD :: "rt ⇒ ip set"
  where "iD rt ≡ {dip. flag rt dip = Some inv}"

definition kD :: "rt ⇒ ip set"
  where "kD rt ≡ {dip. rt dip ≠ None}"

lemma kD_is_vD_and_iD: "kD rt = vD rt ∪ iD rt"
  unfolding kD_def vD_def iD_def by auto

lemma vD_iD_gives_kD [simp]:
  "∧ip rt. ip ∈ vD rt ⇒ ip ∈ kD rt"
  "∧ip rt. ip ∈ iD rt ⇒ ip ∈ kD rt"
  unfolding kD_is_vD_and_iD by simp_all

lemma kD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ kD rt"
  shows "∃ dsn dsk flag hops nhip.
    σroute(rt, dip) = Some (dsn, dsk, flag, hops, nhip)"
  using assms unfolding kD_def by simp

lemma kD_None [dest]:
  fixes dip rt
  assumes "dip ∉ kD rt"
  shows "σroute(rt, dip) = None"
  using assms unfolding kD_def
  by (metis (mono_tags) mem_Collect_eq)

lemma vD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ vD rt"
  shows "∃ dsn dsk hops nhip.
    σroute(rt, dip) = Some (dsn, dsk, val, hops, nhip)"
  using assms unfolding vD_def by simp

lemma vD_empty [simp]: "vD Map.empty = {}"
  unfolding vD_def by simp

lemma iD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ iD rt"
  shows "∃ dsn dsk hops nhip.
    σroute(rt, dip) = Some (dsn, dsk, inv, hops, nhip)"
  using assms unfolding iD_def by simp

lemma val_is_vD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "the (flag rt ip) = val"
  shows "ip ∈ vD(rt)"
  using assms unfolding vD_def by auto

lemma inv_is_iD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "the (flag rt ip) = inv"
  shows "ip ∈ iD(rt)"
  using assms unfolding iD_def by auto

lemma iD_flag_is_inv [elim, simp]:
  fixes ip rt
  assumes "ip ∈ iD(rt)"
  shows "the (flag rt ip) = inv"
  proof -

```

```

    from ⟨ip∈iD(rt)⟩ have "ip∈kD(rt)" by auto
    with assms show ?thesis unfolding iD_def by auto
qed

lemma kD_but_not_vD_is_iD [elim]:
  fixes ip rt
  assumes "ip∈kD(rt)"
    and "ip∉vD(rt)"
  shows "ip∈iD(rt)"
proof -
  from ⟨ip∈kD(rt)⟩ obtain dsn dsk f hops nhop
    where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop)"
    by (metis kD_Some)
  from ⟨ip∉vD(rt)⟩ have "f ≠ val"
  proof (rule contrapos_nn)
    assume "f = val"
    with rtip have "the (flag rt ip) = val" by simp
    with ⟨ip∈kD(rt)⟩ show "ip∈vD(rt)" ..
  qed
  with rtip have "the (flag rt ip) = inv" by simp
  with ⟨ip∈kD(rt)⟩ show "ip∈iD(rt)" ..
qed

lemma vD_or_iD [elim]:
  fixes ip rt
  assumes "ip∈kD(rt)"
    and "ip∈vD(rt) ⇒ P rt ip"
    and "ip∈iD(rt) ⇒ P rt ip"
  shows "P rt ip"
proof -
  from ⟨ip∈kD(rt)⟩ have "ip∈vD(rt) ∪ iD(rt)"
    by (simp add: kD_is_vD_and_iD)
  thus ?thesis by (auto elim: assms(2-3))
qed

lemma proj5_eq_dhops: "∧dip rt. dip∈kD(rt) ⇒ π5(the (rt dip)) = the (dhops rt dip)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj4_eq_flag: "∧dip rt. dip∈kD(rt) ⇒ π4(the (rt dip)) = the (flag rt dip)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj2_eq_sqn: "∧dip rt. dip∈kD(rt) ⇒ π2(the (rt dip)) = sqn rt dip"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 [simp]:
  "∧ip rt. ip∈kD(rt) ⇒ sqnf rt ip = π3(the (rt ip))"
  unfolding sqnf_def by auto

lemma vD_flag_val [simp]:
  "∧dip rt. dip ∈ vD (rt) ⇒ the (flag rt dip) = val"
  unfolding vD_def by clarsimp

lemma kD_update [simp]:
  "∧rt nip v. kD (rt(nip ↦ v)) = insert nip (kD rt)"
  unfolding kD_def by auto

lemma kD_empty [simp]: "kD Map.empty = {}"
  unfolding kD_def by simp

lemma ip_equal_or_known [elim]:
  fixes rt ip ip'
  assumes "ip = ip' ∨ ip∈kD(rt)"
    and "ip = ip' ⇒ P rt ip ip'"
    and "[ ip ≠ ip'; ip∈kD(rt) ] ⇒ P rt ip ip'"

```

```

shows "P rt ip ip'"
using assms by auto

```

### 3.1.4 Updating Routing Tables

Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.

#### Updating route entries

```

lemma in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∈ kD(rt)"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = es (the (rt dip))"
  using assms [THEN kD_Some] by auto

```

```

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∉ kD(rt)"
  shows "(case rt dip of None ⇒ en | Some r ⇒ es r) = en"
  using assms [THEN kD_None] by auto

```

```

lemma rt_Some_sqn [dest]:
  fixes rt and ip dsn dsk flag hops nhip
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip)"
  shows "sqn rt ip = dsn"
  unfolding sqn_def using assms by simp

```

```

lemma not_kD_sqn [simp]:
  fixes dip rt
  assumes "dip ∉ kD(rt)"
  shows "sqn rt dip = 0"
  using assms unfolding sqn_def
  by simp

```

```

definition update_arg_wf :: "r ⇒ bool"
where "update_arg_wf r ≡ π4(r) = val ∧
      (π2(r) = 0) = (π3(r) = unk) ∧
      (π3(r) = unk ⟶ π5(r) = 1)"

```

```

lemma update_arg_wf_gives_cases:
  "∧r. update_arg_wf r ⟹ (π2(r) = 0) = (π3(r) = unk)"
  unfolding update_arg_wf_def by simp

```

```

lemma update_arg_wf_tuples [simp]:
  "∧nhip. update_arg_wf (0, unk, val, Suc 0, nhip)"
  "∧n hops nhip. update_arg_wf (Suc n, kno, val, hops, nhip)"
  unfolding update_arg_wf_def by auto

```

```

lemma update_arg_wf_tuples' [elim]:
  "∧n hops nhip. Suc 0 ≤ n ⟹ update_arg_wf (n, kno, val, hops, nhip)"
  unfolding update_arg_wf_def by auto

```

```

lemma wf_r_cases [intro]:
  fixes P r
  assumes "update_arg_wf r"
  and c1: "∧nhip. P (0, unk, val, Suc 0, nhip)"
  and c2: "∧dsn hops nhip. dsn > 0 ⟹ P (dsn, kno, val, hops, nhip)"
  shows "P r"
proof -
  obtain dsn dsk flag hops nhip
  where *: "r = (dsn, dsk, flag, hops, nhip)" by (cases r)
  with ⟨update_arg_wf r⟩ have wf1: "flag = val"
  and wf2: "(dsn = 0) = (dsk = unk)"

```

```

        and wf3: "dsk = unk  $\longrightarrow$  (hops = 1)"
      unfolding update_arg_wf_def by auto
    have "P (dsn, dsk, flag, hops, nhip)"
    proof (cases dsk)
      assume "dsk = unk"
      moreover with wf2 wf3 have "dsn = 0" and "hops = Suc 0" by auto
      ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c1)
    next
      assume "dsk = kno"
      moreover with wf2 have "dsn > 0" by simp
      ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c2)
    qed
  with * show "P r" by simp
qed

```

definition update :: "rt  $\Rightarrow$  ip  $\Rightarrow$  r  $\Rightarrow$  rt"

```

where
  "update rt ip r  $\equiv$ 
    case  $\sigma_{route}(rt, ip)$  of
      None  $\Rightarrow$  rt (ip  $\mapsto$  r)
    | Some s  $\Rightarrow$ 
      if  $\pi_2(s) < \pi_2(r)$  then rt (ip  $\mapsto$  r)
      else if  $\pi_2(s) = \pi_2(r) \wedge (\pi_5(s) > \pi_5(r) \vee \pi_4(s) = inv)$ 
        then rt (ip  $\mapsto$  r)
      else if  $\pi_3(r) = unk$ 
        then rt (ip  $\mapsto$  ( $\pi_2(s)$ , snd (r)))
      else rt (ip  $\mapsto$  s)"

```

lemma update\_simps [simp]:

```

fixes r s nrt nr' ns rt ip
defines "s  $\equiv$  the  $\sigma_{route}(rt, ip)$ "
  and "nr'  $\equiv$  ( $\pi_2(s)$ ,  $\pi_3(r)$ ,  $\pi_4(r)$ ,  $\pi_5(r)$ ,  $\pi_6(r)$ )"
shows
  "[ip  $\notin$  kD(rt)]  $\implies$  update rt ip r = rt (ip  $\mapsto$  r)"
  "[ip  $\in$  kD(rt); sqn rt ip <  $\pi_2(r)$ ]  $\implies$  update rt ip r = rt (ip  $\mapsto$  r)"
  "[ip  $\in$  kD(rt); sqn rt ip =  $\pi_2(r)$ ;
    the (dhops rt ip) >  $\pi_5(r)$ ]  $\implies$  update rt ip r = rt (ip  $\mapsto$  r)"
  "[ip  $\in$  kD(rt); sqn rt ip =  $\pi_2(r)$ ;
    flag rt ip = Some inv]  $\implies$  update rt ip r = rt (ip  $\mapsto$  r)"
  "[ip  $\in$  kD(rt);  $\pi_3(r) = unk$ ; ( $\pi_2(r) = 0$ ) = ( $\pi_3(r) = unk$ )]  $\implies$  update rt ip r = rt (ip  $\mapsto$  nr)"
  "[ip  $\in$  kD(rt); sqn rt ip  $\geq$   $\pi_2(r)$ ;  $\pi_3(r) = kno$ ;
    sqn rt ip =  $\pi_2(r)$ ]  $\implies$  the (dhops rt ip)  $\leq$   $\pi_5(r) \wedge$  the (flag rt ip) = val ]
     $\implies$  update rt ip r = rt (ip  $\mapsto$  s)"

```

proof -

```

  assume "ip  $\notin$  kD(rt)"
  hence " $\sigma_{route}(rt, ip) = None$ " ..
  thus "update rt ip r = rt (ip  $\mapsto$  r)"
    unfolding update_def by simp
next
  assume "ip  $\in$  kD(rt)"
  and "sqn rt ip <  $\pi_2(r)$ "
  from this(1) obtain dsn dsk fl hops nhip
    where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
    by (metis kD_Some)
  with ⟨sqn rt ip <  $\pi_2(r)$ ⟩ show "update rt ip r = rt (ip  $\mapsto$  r)"
    unfolding update_def s_def by auto
next
  assume "ip  $\in$  kD(rt)"
  and "sqn rt ip =  $\pi_2(r)$ "
  and "the (dhops rt ip) >  $\pi_5(r)$ "
  from this(1) obtain dsn dsk fl hops nhip
    where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
    by (metis kD_Some)
  with ⟨sqn rt ip =  $\pi_2(r)$ ⟩ and ⟨the (dhops rt ip) >  $\pi_5(r)$ ⟩

```

```

  show "update rt ip r = rt (ip ↦ r)"
    unfolding update_def s_def by auto
next
  assume "ip ∈ kD(rt)"
    and "sqn rt ip = π2(r)"
    and "flag rt ip = Some inv"
  from this(1) obtain dsn dsk fl hops nhip
    where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
      by (metis kD_Some)
  with ⟨sqn rt ip = π2(r)⟩ and ⟨flag rt ip = Some inv⟩
  show "update rt ip r = rt (ip ↦ r)"
    unfolding update_def s_def by auto
next
  assume "ip ∈ kD(rt)"
    and "π3(r) = unk"
    and "(π2(r) = 0) = (π3(r) = unk)"
  from this(1) obtain dsn dsk fl hops nhip
    where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
      by (metis kD_Some)
  with ⟨(π2(r) = 0) = (π3(r) = unk)⟩ and ⟨π3(r) = unk⟩
  show "update rt ip r = rt (ip ↦ nr'"
    unfolding update_def nr'_def s_def
      by (cases r) simp
next
  assume "ip ∈ kD(rt)"
    and otherassms: "sqn rt ip ≥ π2(r)"
      "π3(r) = kno"
      "sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val"
  from this(1) obtain dsn dsk fl hops nhip
    where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
      by (metis kD_Some)
  with otherassms show "update rt ip r = rt (ip ↦ s)"
    unfolding update_def s_def by auto
qed

```

lemma update\_cases [elim]:

```

  assumes "(π2(r) = 0) = (π3(r) = unk)"
    and c1: "[ip ∉ kD(rt)] ⇒ P (rt (ip ↦ r))"

    and c2: "[ip ∈ kD(rt); sqn rt ip < π2(r)]
      ⇒ P (rt (ip ↦ r))"
    and c3: "[ip ∈ kD(rt); sqn rt ip = π2(r); the (dhops rt ip) > π5(r)]
      ⇒ P (rt (ip ↦ r))"
    and c4: "[ip ∈ kD(rt); sqn rt ip = π2(r); the (flag rt ip) = inv]
      ⇒ P (rt (ip ↦ r))"
    and c5: "[ip ∈ kD(rt); π3(r) = unk]
      ⇒ P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r),
        π4(r), π5(r), π6(r))))"
    and c6: "[ip ∈ kD(rt); sqn rt ip ≥ π2(r); π3(r) = kno;
      sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val]
      ⇒ P (rt (ip ↦ the σroute(rt, ip)))"

```

shows "(P (update rt ip r))"

proof (cases "ip ∈ kD(rt)")

assume "ip ∉ kD(rt)"

with c1 show ?thesis

by simp

next

assume "ip ∈ kD(rt)"

moreover then obtain dsn dsk fl hops nhip

where rteq: "rt ip = Some (dsn, dsk, fl, hops, nhip)"

by (metis kD\_Some)

moreover obtain dsn' dsk' fl' hops' nhip'

where req: "r = (dsn', dsk', fl', hops', nhip)'"

by (cases r) metis

```

ultimately show ?thesis
using ⟨(π2(r) = 0) = (π3(r) = unk)⟩
  c2 [OF ⟨ip ∈ kD(rt)⟩]
  c3 [OF ⟨ip ∈ kD(rt)⟩]
  c4 [OF ⟨ip ∈ kD(rt)⟩]
  c5 [OF ⟨ip ∈ kD(rt)⟩]
  c6 [OF ⟨ip ∈ kD(rt)⟩]
unfolding update_def sqn_def by auto
qed

lemma update_cases_kD:
assumes "(π2(r) = 0) = (π3(r) = unk)"
  and "ip ∈ kD(rt)"
  and c2: "sqn rt ip < π2(r) ⇒ P (rt (ip ↦ r))"
  and c3: "[sqn rt ip = π2(r); the (dhops rt ip) > π5(r)]
    ⇒ P (rt (ip ↦ r))"
  and c4: "[sqn rt ip = π2(r); the (flag rt ip) = inv]
    ⇒ P (rt (ip ↦ r))"
  and c5: "π3(r) = unk ⇒ P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r),
    π4(r), π5(r), π6(r))))"
  and c6: "[sqn rt ip ≥ π2(r); π3(r) = kno;
    sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val]
    ⇒ P (rt (ip ↦ the σroute(rt, ip)))"
shows "(P (update rt ip r))"
using assms(1) proof (rule update_cases)
  assume "sqn rt ip < π2(r)"
  thus "P (rt(ip ↦ r))" by (rule c2)
next
  assume "sqn rt ip = π2(r)"
  and "the (dhops rt ip) > π5(r)"
  thus "P (rt(ip ↦ r))"
  by (rule c3)
next
  assume "sqn rt ip = π2(r)"
  and "the (flag rt ip) = inv"
  thus "P (rt(ip ↦ r))"
  by (rule c4)
next
  assume "π3(r) = unk"
  thus "P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r), π4(r), π5(r), π6(r))))"
  by (rule c5)
next
  assume "sqn rt ip ≥ π2(r)"
  and "π3(r) = kno"
  and "sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val"
  thus "P (rt (ip ↦ the (rt ip)))"
  by (rule c6)
qed (simp add: ⟨ip ∈ kD(rt)⟩)

```

```

lemma in_kD_after_update [simp]:
fixes rt nip dsn dsk flag hops nhip
shows "kD (update rt nip (dsn, dsk, flag, hops, nhip)) = insert nip (kD rt)"
unfolding update_def
by (cases "rt nip") auto

```

```

lemma nhop_of_update [simp]:
fixes rt dip dsn dsk flag hops nhip
assumes "rt ≠ update rt dip (dsn, dsk, flag, hops, nhip)"
shows "the (nhop (update rt dip (dsn, dsk, flag, hops, nhip)) dip) = nhip"
proof -
from assms
have update_neq: "∧v. rt dip = Some v ⇒
  update rt dip (dsn, dsk, flag, hops, nhip)
  ≠ rt(dip ↦ the (rt dip))"

```

```

    by auto
show ?thesis
proof (cases "rt dip = None")
  assume "rt dip = None"
  thus "?thesis" unfolding update_def by clarsimp
next
  assume "rt dip  $\neq$  None"
  then obtain v where "rt dip = Some v" by (metis not_None_eq)
  with update_neq [OF this] show ?thesis
  unfolding update_def by auto
qed
qed

lemma sqn_if_updated:
  fixes rip v rt ip
  shows "sqn ( $\lambda x. \text{if } x = \text{rip} \text{ then } \text{Some } v \text{ else } \text{rt } x$ ) ip
        = (if ip = rip then  $\pi_2(v)$  else sqn rt ip)"
  unfolding sqn_def by simp

lemma update_sqn [simp]:
  fixes rt dip rip dsn dsk hops nhip
  assumes "(dsn = 0) = (dsk = unk)"
  shows "sqn rt dip  $\leq$  sqn (update rt rip (dsn, dsk, val, hops, nhip)) dip"
  proof (rule update_cases)
    show "( $\pi_2$  (dsn, dsk, val, hops, nhip) = 0) = ( $\pi_3$  (dsn, dsk, val, hops, nhip) = unk)"
    by simp (rule assms)
  qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
  fixes rt ip ip' dsn dsk flag hops nhip
  assumes "1  $\leq$  hops"
  shows "sqn rt ip  $\leq$  sqn (update rt ip' (dsn, dsk, flag, hops, nhip)) ip"
  using assms unfolding update_def sqn_def
  by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
  fixes rt dsn dsk flag hops ip rip nhip
  assumes ex: " $\forall ip \in kD \text{ rt. the } (dhops \text{ rt } ip) \geq 1$ "
    and ip: "(ip = rip  $\wedge$  Suc 0  $\leq$  hops)  $\vee$  (ip  $\neq$  rip  $\wedge$  ip  $\in kD \text{ rt})$ "
  shows "Suc 0  $\leq$  the (dhops (update rt rip (dsn, dsk, flag, hops, nhip)) ip)"
  using ip proof
    assume "ip = rip  $\wedge$  Suc 0  $\leq$  hops" thus ?thesis
    unfolding update_def using ex
    by (cases "rip  $\in kD \text{ rt}")$  (drule(1) bspec, auto)
  next
    assume "ip  $\neq$  rip  $\wedge$  ip  $\in kD \text{ rt}"$  thus ?thesis
    using ex unfolding update_def
    by (cases "rip  $\in kD \text{ rt}")$  auto
  qed

lemma update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip  $\neq$  dip"
  shows "(update rt dip (dsn, dsk, flag, hops, nhip)) ip = rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma nhop_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip  $\neq$  dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nhip)) ip = nhop rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

```



```

lemma dhops_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip ≠ dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip)) ip = dhops rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
  "∧rt ip dsn dsk flag hops nhip. sqn (rt(ip ↦ v)) ip = π2(v)"
  unfolding sqn_def by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip
  assumes "rt ≠ update rt dip (osn, kno, val, hops, nhip)"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nhip)) dip) = hops"
  using assms unfolding update_def
  by (clarsimp split: option.split_asm option.split split_if_asm) auto

lemma nhop_update_unk_val [simp]:
  "∧rt dip ip dsn hops.
  the (nhop (update rt dip (dsn, unk, val, hops, ip)) dip) = ip"
  unfolding update_def by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
  assumes "update rt dip (dsn, dsk, flg, hops, sip) ≠ rt"
  shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip)) dip) = sip"
  using assms unfolding update_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma update_rt_split_asm:
  "∧rt ip dsn dsk flag hops sip.
  P (update rt ip (dsn, dsk, flag, hops, sip))
  =
  (¬(rt = update rt ip (dsn, dsk, flag, hops, sip)) ∧ ¬P rt
  ∨ rt ≠ update rt ip (dsn, dsk, flag, hops, sip)
  ∧ ¬P (update rt ip (dsn, dsk, flag, hops, sip)))"
  by auto

lemma sqn_update [simp]: "∧rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip)
  ⇒ sqn (update rt dip (dsn, kno, flg, hops, sip)) dip = dsn"
  unfolding update_def by (clarsimp split: option.split split_if_asm) auto

lemma sqnf_update [simp]: "∧rt dip dsn dsk flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip)
  ⇒ sqnf (update rt dip (dsn, dsk, flg, hops, sip)) dip = dsk"
  unfolding update_def sqnf_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma update_kno_dsn_greater_zero:
  "∧rt dip ip dsn hops. 1 ≤ dsn ⇒ 1 ≤ (sqn (update rt dip (dsn, kno, val, hops, ip)) dip)"
  unfolding update_def
  by (clarsimp split: option.splits)

lemma proj3_update [simp]: "∧rt dip dsn dsk flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip)
  ⇒ π3(the (update rt dip (dsn, dsk, flg, hops, sip) dip)) = dsk"
  unfolding update_def sqnf_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma nhop_update_changed_kno_val [simp]: "∧rt ip dsn dsk hops nhip.
  rt ≠ update rt ip (dsn, kno, val, hops, nhip)
  ⇒ the (nhop (update rt ip (dsn, kno, val, hops, nhip)) ip) = nhip"

```

```

using assms unfolding update_def
by (clarsimp split: option.split_asm option.split_split_if_asm) auto

lemma flag_update [simp]: "\rt dip dsn flg hops sip.
rt ≠ update rt dip (dsn, kno, flg, hops, sip)
⇒ the (flag (update rt dip (dsn, kno, flg, hops, sip)) dip) = flg"
unfolding update_def
by (clarsimp split: option.split_split_if_asm) auto

lemma the_flag_Some [dest!]:
  fixes ip rt
  assumes "the (flag rt ip) = x"
  and "ip ∈ kD rt"
  shows "flag rt ip = Some x"
using assms by auto

lemma kD_update_unchanged [dest]:
  fixes rt dip dsn dsk flag hops nhip
  assumes "rt = update rt dip (dsn, dsk, flag, hops, nhip)"
  shows "dip ∈ kD(rt)"
proof -
  have "dip ∈ kD(update rt dip (dsn, dsk, flag, hops, nhip))" by simp
  with assms show ?thesis by simp
qed

lemma nhop_update [simp]: "\rt dip dsn dsk flg hops sip.
rt ≠ update rt dip (dsn, dsk, flg, hops, sip)
⇒ the (nhop (update rt dip (dsn, dsk, flg, hops, sip)) dip) = sip"
unfolding update_def sqnf_def
by (clarsimp split: option.splits_split_if_asm) auto

lemma sqn_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip ≠ dip"
  shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip)) ip = sqn rt ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.splits) auto

lemma sqnf_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip ≠ dip"
  shows "sqnf (update rt dip (dsn, dsk, flag, hops, nhip)) ip = sqnf rt ip"
using assms unfolding update_def sqnf_def
by (clarsimp split: option.splits) auto

lemma vD_update_val [dest]:
  "\dip rt dip' dsn dsk hops nhip.
dip ∈ vD(update rt dip' (dsn, dsk, val, hops, nhip)) ⇒ (dip ∈ vD(rt) ∨ dip=dip'"
unfolding update_def vD_def by (clarsimp split: option.split_asm split_if_asm)

```

## Invalidating route entries

```

definition invalidate :: "rt ⇒ (ip → sqn) ⇒ rt"
where "invalidate rt dests ≡
λip. case (rt ip, dests ip) of
  (None, _) ⇒ None
| (Some s, None) ⇒ Some s
| (Some (_, dsk, _, hops, nhip), Some rsn) ⇒
  Some (rsn, dsk, inv, hops, nhip)"

```

```

lemma proj3_invalidate [simp]:
  "\dip. π3(the ((invalidate rt dests) dip)) = π3(the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)

```

```

lemma proj5_invalidate [simp]:
  " $\bigwedge$ dip.  $\pi_5(\text{the}((\text{invalidate } \text{rt } \text{dests}) \text{ dip})) = \pi_5(\text{the}(\text{rt } \text{dip}))"$ 
  unfolding invalidate_def by (clarsimp split: option.split)

lemma proj6_invalidate [simp]:
  " $\bigwedge$ dip.  $\pi_6(\text{the}((\text{invalidate } \text{rt } \text{dests}) \text{ dip})) = \pi_6(\text{the}(\text{rt } \text{dip}))"$ 
  unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_kD_inv [simp]:
  " $\bigwedge$ rt dests.  $\text{kD}(\text{invalidate } \text{rt } \text{dests}) = \text{kD } \text{rt}"$ 
  unfolding invalidate_def kD_def
  by (simp split: option.split)

lemma invalidate_sqn:
  fixes rt dip dests
  assumes " $\forall$ rsn.  $\text{dests } \text{dip} = \text{Some } \text{rsn} \longrightarrow \text{sqn } \text{rt } \text{dip} \leq \text{rsn}"$ 
  shows " $\text{sqn } \text{rt } \text{dip} \leq \text{sqn}(\text{invalidate } \text{rt } \text{dests}) \text{ dip}"$ 
  proof (cases "dip  $\notin$   $\text{kD}(\text{rt})"$ )
    assume " $\neg$  dip  $\notin$   $\text{kD}(\text{rt})"$ "
    hence "dip  $\in$   $\text{kD}(\text{rt})"$  by simp
    then obtain dsn dsk flag hops nhip pre where "rt dip = Some (dsn, dsk, flag, hops, nhip)"
      by (metis kD_Some)
    with assms show " $\text{sqn } \text{rt } \text{dip} \leq \text{sqn}(\text{invalidate } \text{rt } \text{dests}) \text{ dip}"$ 
      by (cases "dests dip") (auto simp add: invalidate_def sqn_def)
  qed simp

lemma sqn_invalidate_in_dests [simp]:
  fixes dests ipa rsn rt
  assumes "dests ipa = Some rsn"
    and "ipa  $\in$   $\text{kD}(\text{rt})"$ 
  shows " $\text{sqn}(\text{invalidate } \text{rt } \text{dests}) \text{ ipa} = \text{rsn}"$ 
  unfolding invalidate_def sqn_def
  using assms(1) assms(2) [THEN kD_Some]
  by clarsimp

lemma dhops_invalidate [simp]:
  " $\bigwedge$ dip.  $\text{the}(\text{dhops}(\text{invalidate } \text{rt } \text{dests}) \text{ dip}) = \text{the}(\text{dhops } \text{rt } \text{dip})"$ 
  unfolding invalidate_def by (clarsimp split: option.split)

lemma sqnf_invalidate [simp]:
  " $\bigwedge$ dip.  $\text{sqnf}(\text{invalidate } \text{rt } \xi)(\text{dests } \xi) \text{ dip} = \text{sqnf}(\text{rt } \xi) \text{ dip}"$ 
  unfolding sqnf_def invalidate_def by (clarsimp split: option.split)

lemma nhop_invalidate [simp]:
  " $\bigwedge$ dip.  $\text{the}(\text{nhop}(\text{invalidate } \text{rt } \xi)(\text{dests } \xi) \text{ dip}) = \text{the}(\text{nhop } \text{rt } \xi) \text{ dip}"$ 
  unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
  fixes rt dests dip
  assumes "dip  $\notin$   $\text{dom}(\text{dests})"$ 
  shows " $\text{invalidate } \text{rt } \text{dests } \text{dip} = \text{rt } \text{dip}"$ 
  using assms unfolding invalidate_def
  by (clarsimp split: option.split_asm)

lemma invalidate_none [simp]:
  fixes rt dests dip
  assumes "dip  $\notin$   $\text{kD}(\text{rt})"$ 
  shows " $\text{invalidate } \text{rt } \text{dests } \text{dip} = \text{None}"$ 
  using assms unfolding invalidate_def by clarsimp

lemma vD_invalidate_vD_not_dests:
  " $\bigwedge$ dip rt dests.  $\text{dip} \in \text{vD}(\text{invalidate } \text{rt } \text{dests}) \implies \text{dip} \in \text{vD}(\text{rt}) \wedge \text{dests } \text{dip} = \text{None}"$ 
  unfolding invalidate_def vD_def

```

```

by (clarsimp split: option.split_asm)

lemma sqn_invalidate_not_in_dests [simp]:
  fixes dests dip rt
  assumes "dip ∉ dom(dests)"
  shows "sqn (invalidate rt dests) dip = sqn rt dip"
  using assms unfolding sqn_def by simp

lemma invalidate_changes:
  fixes rt dests dip dsn dsk flag hops nhip
  assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip)"
  shows " dsn = (case dests dip of None ⇒ π2(the (rt dip)) | Some rsn ⇒ rsn)
    ∧ dsk = π3(the (rt dip))
    ∧ flag = (if dests dip = None then π4(the (rt dip)) else inv)
    ∧ hops = π5(the (rt dip))
    ∧ nhip = π6(the (rt dip))"
  using assms unfolding invalidate_def
  by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: "∧dip rt dests. dip ∈ kD (rt)
  ⇒ π3(the (invalidate rt dests dip)) = π3(the (rt dip))"
  by (clarsimp simp: invalidate_def kD_def split: option.split)

lemma dests_iD_invalidate [simp]:
  assumes "dests ip = Some rsn"
  and "ip ∈ kD(rt)"
  shows "ip ∈ iD(invalidate rt dests)"
  using assms(1) assms(2) [THEN kD_Some] unfolding invalidate_def iD_def
  by (clarsimp split: option.split)

```

### 3.1.5 Route Requests

Generate a fresh route request identifier.

```

definition nrreqid :: "(ip × rreqid) set ⇒ ip ⇒ rreqid"
  where "nrreqid rreqs ip ≡ Max ({n. (ip, n) ∈ rreqs} ∪ {0}) + 1"

```

### 3.1.6 Queued Packets

Functions for sending data packets.

```

type_synonym store = "ip → (p × data list)"

```

```

definition sigma_queue :: "store ⇒ ip ⇒ data list" ("σqueue'(_, _)'")
  where "σqueue(store, dip) ≡ case store dip of None ⇒ [] | Some (p, q) ⇒ q"

```

```

definition qD :: "store ⇒ ip set"
  where "qD ≡ dom"

```

```

definition add :: "data ⇒ ip ⇒ store ⇒ store"
  where "add d dip store ≡ case store dip of
    None ⇒ store (dip ↦ (req, [d]))
    | Some (p, q) ⇒ store (dip ↦ (p, q @ [d]))"

```

```

lemma qD_add [simp]:
  fixes d dip store
  shows "qD(add d dip store) = insert dip (qD store)"
  unfolding add_def Let_def qD_def
  by (clarsimp split: option.split)

```

```

definition drop :: "ip ⇒ store → store"
  where "drop dip store ≡
  map_option (λ(p, q). if tl q = [] then store (dip := None)
    else store (dip ↦ (p, tl q))) (store dip)"

```

```

definition sigma_p_flag :: "store ⇒ ip → p" ("σp-flag'(, -)")
  where "σp-flag(store, dip) ≡ map_option fst (store dip)"

```

```

definition unsetRRF :: "store ⇒ ip ⇒ store"
  where "unsetRRF store dip ≡ case store dip of
    None ⇒ store
    | Some (p, q) ⇒ store (dip ↦ (noreq, q))"

```

```

definition setRRF :: "store ⇒ (ip → sqn) ⇒ store"
  where "setRRF store dests ≡ λdip. if dests dip = None then store dip
    else map_option (λ(, q). (req, q)) (store dip)"

```

### 3.1.7 Comparison with the original technical report

The major differences with the AODV technical report of Fehnker et al are:

1. *nhop* is partial, thus a ‘*the*’ is needed, similarly for *dhops* and *addpreRT*.
2. *precs* is partial.
3.  $\sigma_{p\text{-flag}}(\text{store}, \text{dip})$  is partial.
4. The routing table (*rt*) is modelled as a map ( $\text{ip} \Rightarrow r \text{ option}$ ) rather than a set of 7-tuples, likewise, the *r* is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address (*dip*) is taken from the argument to the function, rather than a part of the result. Well-definedness then follows from the structure of the type and more related facts are available automatically, rather than having to be acquired through tedious proofs.
5. Similar remarks hold for the *dests* mapping passed to *invalidate*, and *store*.

end

## 3.2 AODV protocol messages

```

theory C_Aodv_Message
imports C_Gtobcast
begin

```

```

datatype msg =
  Rreq nat rreqid ip sqn k ip sqn ip
  | Rrep nat ip sqn ip ip
  | Rerr "ip → sqn" ip
  | Newpkt data ip
  | Pkt data ip ip

```

```

instantiation msg :: msg

```

```

begin

```

```

  definition newpkt_def [simp]: "newpkt ≡ λ(d, dip). Newpkt d dip"

```

```

  definition eq_newpkt_def: "eq_newpkt m ≡ case m of Newpkt d dip ⇒ True | _ ⇒ False"

```

```

  instance by intro_classes (simp add: eq_newpkt_def)

```

```

end

```

The *msg* type models the different messages used within AODV. The instantiation as a *msg* is a technicality due to the special treatment of *newpkt* messages in the AWN SOS rules. This use of classes allows a clean separation of the AWN-specific definitions and these AODV-specific definitions.

```

definition rreq :: "nat × rreqid × ip × sqn × k × ip × sqn × ip ⇒ msg"
  where "rreq ≡ λ(hops, rreqid, dip, dsn, dsk, oip, osn, sip).
    Rreq hops rreqid dip dsn dsk oip osn sip"

```

```

lemma rreq_simp [simp]:

```

```

  "rreq(hops, rreqid, dip, dsn, dsk, oip, osn, sip) = Rreq hops rreqid dip dsn dsk oip osn sip"
  unfolding rreq_def by simp

```

```

definition rrep :: "nat × ip × sqn × ip × ip ⇒ msg"
  where "rrep ≡ λ(hops, dip, dsn, oip, sip). Rrep hops dip dsn oip sip"

lemma rrep_simp [simp]:
  "rrep(hops, dip, dsn, oip, sip) = Rrep hops dip dsn oip sip"
  unfolding rrep_def by simp

definition rerr :: "(ip → sqn) × ip ⇒ msg"
  where "rerr ≡ λ(dests, sip). Rerr dests sip"

lemma rerr_simp [simp]:
  "rerr(dests, sip) = Rerr dests sip"
  unfolding rerr_def by simp

lemma not_eq_newpkt_rreq [simp]: "¬eq_newpkt (Rreq hops rreqid dip dsn dsk oip osn sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_rrep [simp]: "¬eq_newpkt (Rrep hops dip dsn oip sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_rerr [simp]: "¬eq_newpkt (Rerr dests sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_pkt [simp]: "¬eq_newpkt (Pkt d dip sip)"
  unfolding eq_newpkt_def by simp

definition pkt :: "data × ip × ip ⇒ msg"
  where "pkt ≡ λ(d, dip, sip). Pkt d dip sip"

lemma pkt_simp [simp]:
  "pkt(d, dip, sip) = Pkt d dip sip"
  unfolding pkt_def by simp

```

end

### 3.3 The AODV protocol

```

theory C_Aodv
imports C_Aodv_Data C_Aodv_Message
  "../.../AWN/AWN_SOS_Labels" "../.../AWN/AWN_Invariants"
begin

```

#### 3.3.1 Data state

```

record state =
  ip      :: "ip"
  sn      :: "sqn"
  rt      :: "rt"
  rreqs   :: "(ip × rreqid) set"
  store   :: "store"

  msg     :: "msg"
  data    :: "data"
  dests   :: "ip → sqn"
  rreqid  :: "rreqid"
  dip     :: "ip"
  oip     :: "ip"
  hops    :: "nat"
  dsn     :: "sqn"
  dsk     :: "k"
  osn     :: "sqn"
  sip     :: "ip"

```

abbreviation aadv\_init :: "ip  $\Rightarrow$  state"

```
where "aadv_init i  $\equiv$  (  
  ip = i,  
  sn = 1,  
  rt = empty,  
  rreqs = {},  
  store = empty,  
  
  msg    = (SOME x. True),  
  data   = (SOME x. True),  
  dests  = (SOME x. True),  
  rreqid = (SOME x. True),  
  dip    = (SOME x. True),  
  oip    = (SOME x. True),  
  hops   = (SOME x. True),  
  dsn    = (SOME x. True),  
  dsk    = (SOME x. True),  
  osn    = (SOME x. True),  
  sip    = (SOME x. x  $\neq$  i)  
)"
```

lemma some\_neq\_not\_eq [simp]: " $\neg((\text{SOME } x :: \text{nat. } x \neq i) = i)$ "  
by (subst some\_eq\_ex) (metis zero\_neq\_numeral)

definition clear\_locals :: "state  $\Rightarrow$  state"

```
where "clear_locals  $\xi$  =  $\xi$  (  
  msg    := (SOME x. True),  
  data   := (SOME x. True),  
  dests  := (SOME x. True),  
  rreqid := (SOME x. True),  
  dip    := (SOME x. True),  
  oip    := (SOME x. True),  
  hops   := (SOME x. True),  
  dsn    := (SOME x. True),  
  dsk    := (SOME x. True),  
  osn    := (SOME x. True),  
  sip    := (SOME x. x  $\neq$  ip  $\xi$ )  
)"
```

lemma clear\_locals\_sip\_not\_ip [simp]: " $\neg(\text{sip } (\text{clear\_locals } \xi) = \text{ip } \xi)$ "  
unfolding clear\_locals\_def by simp

lemma clear\_locals\_but\_not\_globals [simp]:

```
"ip (clear_locals  $\xi$ ) = ip  $\xi$ "  
"sn (clear_locals  $\xi$ ) = sn  $\xi$ "  
"rt (clear_locals  $\xi$ ) = rt  $\xi$ "  
"rreqs (clear_locals  $\xi$ ) = rreqs  $\xi$ "  
"store (clear_locals  $\xi$ ) = store  $\xi$ "  
unfolding clear_locals_def by auto
```

### 3.3.2 Auxilliary message handling definitions

definition is\_newpkt

```
where "is_newpkt  $\xi \equiv$  case msg  $\xi$  of  
  Newpkt data' dip'  $\Rightarrow$  {  $\xi$ (data := data', dip := dip') }  
  | _  $\Rightarrow$  {}"
```

definition is\_pkt

```
where "is_pkt  $\xi \equiv$  case msg  $\xi$  of  
  Pkt data' dip' oip'  $\Rightarrow$  {  $\xi$ (data := data', dip := dip', oip := oip' ) }  
  | _  $\Rightarrow$  {}"
```

definition is\_rreq

```
where "is_rreq  $\xi \equiv$  case msg  $\xi$  of
```

```

Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' ⇒
  { ξ(| hops := hops', rreqid := rreqid', dip := dip', dsn := dsn',
      dsk := dsk', oip := oip', osn := osn', sip := sip' |) }
| _ ⇒ {}"

```

lemma *is\_rreq\_asm* [*dest!*]:

```

assumes "ξ' ∈ is_rreq ξ"
shows "(∃ hops' rreqid' dip' dsn' dsk' oip' osn' sip'.
  msg ξ = Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' ∧
  ξ' = ξ(| hops := hops', rreqid := rreqid', dip := dip', dsn := dsn',
      dsk := dsk', oip := oip', osn := osn', sip := sip' |))"
using assms unfolding is_rreq_def
by (cases "msg ξ") simp_all

```

definition *is\_rrep*

```

where "is_rrep ξ ≡ case msg ξ of
  Rrep hops' dip' dsn' oip' sip' ⇒
    { ξ(| hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' |) }
| _ ⇒ {}"

```

lemma *is\_rrep\_asm* [*dest!*]:

```

assumes "ξ' ∈ is_rrep ξ"
shows "(∃ hops' dip' dsn' oip' sip'.
  msg ξ = Rrep hops' dip' dsn' oip' sip' ∧
  ξ' = ξ(| hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' |))"
using assms unfolding is_rrep_def
by (cases "msg ξ") simp_all

```

definition *is\_rerr*

```

where "is_rerr ξ ≡ case msg ξ of
  Rerr dests' sip' ⇒ { ξ(| dests := dests', sip := sip' |) }
| _ ⇒ {}"

```

lemma *is\_rerr\_asm* [*dest!*]:

```

assumes "ξ' ∈ is_rerr ξ"
shows "(∃ dests' sip'.
  msg ξ = Rerr dests' sip' ∧
  ξ' = ξ(| dests := dests', sip := sip' |))"
using assms unfolding is_rerr_def
by (cases "msg ξ") simp_all

```

lemmas *is\_msg\_defs* =

```

is_rerr_def is_rrep_def is_rreq_def is_pkt_def is_newpkt_def

```

lemma *is\_msg\_inv\_ip* [*simp*]:

```

"ξ' ∈ is_rerr ξ ⇒ ip ξ' = ip ξ"
"ξ' ∈ is_rrep ξ ⇒ ip ξ' = ip ξ"
"ξ' ∈ is_rreq ξ ⇒ ip ξ' = ip ξ"
"ξ' ∈ is_pkt ξ ⇒ ip ξ' = ip ξ"
"ξ' ∈ is_newpkt ξ ⇒ ip ξ' = ip ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp)+

```

lemma *is\_msg\_inv\_sn* [*simp*]:

```

"ξ' ∈ is_rerr ξ ⇒ sn ξ' = sn ξ"
"ξ' ∈ is_rrep ξ ⇒ sn ξ' = sn ξ"
"ξ' ∈ is_rreq ξ ⇒ sn ξ' = sn ξ"
"ξ' ∈ is_pkt ξ ⇒ sn ξ' = sn ξ"
"ξ' ∈ is_newpkt ξ ⇒ sn ξ' = sn ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp)+

```

lemma *is\_msg\_inv\_rt* [*simp*]:

```

"ξ' ∈ is_rerr ξ ⇒ rt ξ' = rt ξ"

```



```

" $\xi'$   $\in$  is_rrep  $\xi$   $\implies$  rt  $\xi'$  = rt  $\xi$ "
" $\xi'$   $\in$  is_rreq  $\xi$   $\implies$  rt  $\xi'$  = rt  $\xi$ "
" $\xi'$   $\in$  is_pkt  $\xi$   $\implies$  rt  $\xi'$  = rt  $\xi$ "
" $\xi'$   $\in$  is_newpkt  $\xi$   $\implies$  rt  $\xi'$  = rt  $\xi$ "
unfolding is_msg_defs
by (cases "msg  $\xi$ ", clarsimp+)+

```

```

lemma is_msg_inv_rreqs [simp]:
" $\xi'$   $\in$  is_rerr  $\xi$   $\implies$  rreqs  $\xi'$  = rreqs  $\xi$ "
" $\xi'$   $\in$  is_rrep  $\xi$   $\implies$  rreqs  $\xi'$  = rreqs  $\xi$ "
" $\xi'$   $\in$  is_rreq  $\xi$   $\implies$  rreqs  $\xi'$  = rreqs  $\xi$ "
" $\xi'$   $\in$  is_pkt  $\xi$   $\implies$  rreqs  $\xi'$  = rreqs  $\xi$ "
" $\xi'$   $\in$  is_newpkt  $\xi$   $\implies$  rreqs  $\xi'$  = rreqs  $\xi$ "
unfolding is_msg_defs
by (cases "msg  $\xi$ ", clarsimp+)+

```

```

lemma is_msg_inv_store [simp]:
" $\xi'$   $\in$  is_rerr  $\xi$   $\implies$  store  $\xi'$  = store  $\xi$ "
" $\xi'$   $\in$  is_rrep  $\xi$   $\implies$  store  $\xi'$  = store  $\xi$ "
" $\xi'$   $\in$  is_rreq  $\xi$   $\implies$  store  $\xi'$  = store  $\xi$ "
" $\xi'$   $\in$  is_pkt  $\xi$   $\implies$  store  $\xi'$  = store  $\xi$ "
" $\xi'$   $\in$  is_newpkt  $\xi$   $\implies$  store  $\xi'$  = store  $\xi$ "
unfolding is_msg_defs
by (cases "msg  $\xi$ ", clarsimp+)+

```

```

lemma is_msg_inv_sip [simp]:
" $\xi'$   $\in$  is_pkt  $\xi$   $\implies$  sip  $\xi'$  = sip  $\xi$ "
" $\xi'$   $\in$  is_newpkt  $\xi$   $\implies$  sip  $\xi'$  = sip  $\xi$ "
unfolding is_msg_defs
by (cases "msg  $\xi$ ", clarsimp+)+

```

### 3.3.3 The protocol process

```

datatype pseqp =
  PAadv
  | PNewPkt
  | PPkt
  | PRreq
  | PRrep
  | PRerr

```

```

fun nat_of_seqp :: "pseqp  $\Rightarrow$  nat"
where
  "nat_of_seqp PAadv = 1"
| "nat_of_seqp PPkt = 2"
| "nat_of_seqp PNewPkt = 3"
| "nat_of_seqp PRreq = 4"
| "nat_of_seqp PRrep = 5"
| "nat_of_seqp PRerr = 6"

```

```

instantiation "pseqp" :: ord
begin
definition less_eq_seqp [iff]: " $l1 \leq l2 = (\text{nat\_of\_seqp } l1 \leq \text{nat\_of\_seqp } l2)$ "
definition less_seqp [iff]: " $l1 < l2 = (\text{nat\_of\_seqp } l1 < \text{nat\_of\_seqp } l2)$ "
instance ..
end

```

```

abbreviation AODV
where
  "AODV  $\equiv$   $\lambda$ _. [[clear_locals]] call(PAadv)"

```

```

abbreviation PKT
where
  "PKT args  $\equiv$ 

```

```

[[ξ. let (data, dip, oip) = args ξ in
  (clear_locals ξ) (| data := data, dip := dip, oip := oip |)]
call(PPkt)"
abbreviation NEWPKT
where
  "NEWPKT args ≡
[[ξ. let (data, dip) = args ξ in
  (clear_locals ξ) (| data := data, dip := dip |)]
call(PNewPkt)"

abbreviation RREQ
where
  "RREQ args ≡
[[ξ. let (hops, rreqid, dip, dsn, dsk, oip, osn, sip) = args ξ in
  (clear_locals ξ) (| hops := hops, rreqid := rreqid, dip := dip,
    dsn := dsn, dsk := dsk, oip := oip,
    osn := osn, sip := sip |)]
call(PRreq)"

abbreviation RREP
where
  "RREP args ≡
[[ξ. let (hops, dip, dsn, oip, sip) = args ξ in
  (clear_locals ξ) (| hops := hops, dip := dip, dsn := dsn,
    oip := oip, sip := sip |)]
call(PRrep)"

abbreviation RERR
where
  "RERR args ≡
[[ξ. let (dests, sip) = args ξ in
  (clear_locals ξ) (| dests := dests, sip := sip |)]
call(PRerr)"

fun ΓAODV :: "(state, msg, pseqp, pseqp label) seqp_env"
where
  "ΓAODV PAodv = labelled PAodv (
  receive(λmsg' ξ. ξ (| msg := msg' |)).
  (
    ⟨is_newpkt⟩ NEWPKT(λξ. (data ξ, ip ξ))
    ⊕ ⟨is_pkt⟩ PKT(λξ. (data ξ, dip ξ, oip ξ))
    ⊕ ⟨is_rreq⟩
      [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ) |)]
      RREQ(λξ. (hops ξ, rreqid ξ, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, sip ξ))
    ⊕ ⟨is_rrep⟩
      [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ) |)]
      RREP(λξ. (hops ξ, dip ξ, dsn ξ, oip ξ, sip ξ))
    ⊕ ⟨is_rerr⟩
      [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ) |)]
      RERR(λξ. (dests ξ, sip ξ))
  )
  ⊕ ⟨λξ. { ξ(| dip := dip |) | dip. dip ∈ qD(store ξ) ∩ vD(rt ξ) }⟩
    [[ξ. ξ (| data := hd(σqueue(store ξ, dip ξ)) |)]
    unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, ip ξ)).
    [[ξ. ξ (| store := the (drop (dip ξ) (store ξ)) |)]
    AODV()
    ▷ [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
      then Some (inc (sqn (rt ξ) rip)) else None) |)]
      [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]
      [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]
      broadcast(λξ. rerr(dests ξ, ip ξ)). AODV()
  ⊕ ⟨λξ. { ξ(| dip := dip |)
    | dip. dip ∈ qD(store ξ) - vD(rt ξ) ∧ the (σp-flag(store ξ, dip)) = req }⟩
    [[ξ. ξ (| store := unsetRRF (store ξ) (dip ξ) |)]

```

```

[[ξ. ξ (| sn := inc (sn ξ) )]]
[[ξ. ξ (| rreqid := nrreqid (rreqs ξ) (ip ξ) )]]
[[ξ. ξ (| rreqs := rreqs ξ ∪ {(ip ξ, rreqid ξ)} )]]
broadcast(λξ. rreq(0, rreqid ξ, dip ξ, sqn (rt ξ) (dip ξ), sqnf (rt ξ) (dip ξ), ip ξ, sn ξ,
ip ξ)). AODV()"

```

```

| "ΓAODV PNewPkt = labelled PNewPkt (
  ⟨ξ. dip ξ = ip ξ⟩
  deliver(λξ. data ξ).AODV()
⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
  [[ξ. ξ (| store := add (data ξ) (dip ξ) (store ξ) )]]
  AODV()"

```

```

| "ΓAODV PPkt = labelled PPkt (
  ⟨ξ. dip ξ = ip ξ⟩
  deliver(λξ. data ξ).AODV()
⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
  (
    ⟨ξ. dip ξ ∈ vD (rt ξ)⟩
    unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, oip ξ)).AODV()
  ▷
    [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
      then Some (inc (sqn (rt ξ) rip)) else None) )]]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
    broadcast(λξ. rerr(dests ξ, ip ξ)).AODV()
  ⊕ ⟨ξ. dip ξ ∉ vD (rt ξ)⟩
  (
    ⟨ξ. dip ξ ∈ iD (rt ξ)⟩
    broadcast(λξ. rerr([dip ξ ↦ sqn (rt ξ) (dip ξ)], ip ξ)). AODV()
  ⊕ ⟨ξ. dip ξ ∉ iD (rt ξ)⟩
    AODV()
  )
  )
  )"

```

```

| "ΓAODV PRreq = labelled PRreq (
  ⟨ξ. (oip ξ, rreqid ξ) ∈ rreqs ξ⟩
  AODV()
⊕ ⟨ξ. (oip ξ, rreqid ξ) ∉ rreqs ξ⟩
  [[ξ. ξ (| rt := update (rt ξ) (oip ξ) (osn ξ, kno, val, hops ξ + 1, sip ξ) )]]
  [[ξ. ξ (| rreqs := rreqs ξ ∪ {(oip ξ, rreqid ξ)} )]]
  (
    ⟨ξ. dip ξ = ip ξ⟩
    [[ξ. ξ (| sn := max (sn ξ) (dsn ξ) )]]
    unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(0, dip ξ, sn ξ, oip ξ, ip ξ)).AODV()
  ▷
    [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
      then Some (inc (sqn (rt ξ) rip)) else None) )]]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
    broadcast(λξ. rerr(dests ξ, ip ξ)).AODV()
  ⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
  (
    ⟨ξ. dip ξ ∈ vD (rt ξ) ∧ dsn ξ ≤ sqn (rt ξ) (dip ξ) ∧ sqnf (rt ξ) (dip ξ) = kno⟩
    unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(the (dhops (rt ξ) (dip ξ)), dip ξ,
      sqn (rt ξ) (dip ξ), oip ξ, ip ξ)).
    AODV()
  ▷
    [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
      then Some (inc (sqn (rt ξ) rip)) else None) )]]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
    broadcast(λξ. rerr(dests ξ, ip ξ)).AODV()
  ⊕ ⟨ξ. dip ξ ∉ vD (rt ξ) ∨ sqn (rt ξ) (dip ξ) < dsn ξ ∨ sqnf (rt ξ) (dip ξ) = unk⟩

```

```

        broadcast(λξ. rreq(hops ξ + 1, rreqid ξ, dip ξ, max (sqn (rt ξ) (dip ξ)) (dsn ξ),
                          dsk ξ, oip ξ, osn ξ, ip ξ)).
    AODV()
  )
))"

/ "ΓAODV PRrep = labelled PRrep (
  ⟨ξ. rt ξ ≠ update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ) ⟩
  (
    [[ξ. ξ (| rt := update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ) |)]]
    (
      ⟨ξ. oip ξ = ip ξ ⟩
      AODV()
      ⊕ ⟨ξ. oip ξ ≠ ip ξ ⟩
      (
        ⟨ξ. oip ξ ∈ vD (rt ξ) ⟩
        unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(hops ξ + 1, dip ξ,
              dsn ξ, oip ξ, ip ξ)).
        AODV()
      ▷
        [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
              then Some (inc (sqn (rt ξ) rip)) else None) |)]]
        [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]]
        [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]]
        broadcast(λξ. rerr(dests ξ, ip ξ)).AODV()
      ⊕ ⟨ξ. oip ξ ∉ vD (rt ξ) ⟩
      AODV()
    )
  )
)
)
⊕ ⟨ξ. rt ξ = update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ) ⟩
  AODV()
)"

/ "ΓAODV PRerr = labelled PRerr (
  [[ξ. ξ (| dests := (λrip. case (dests ξ) rip of None ⇒ None
    | Some rsn ⇒ if rip ∈ vD (rt ξ) ∧ the (nhop (rt ξ) rip) = sip ξ
    ∧ sqn (rt ξ) rip < rsn then Some rsn else None) |)]]
  [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]]
  [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]]
  (
    ⟨ξ. dests ξ ≠ Map.empty ⟩
    broadcast(λξ. rerr(dests ξ, ip ξ)). AODV()
    ⊕ ⟨ξ. dests ξ = Map.empty ⟩
    AODV()
  ))"

```

**declare**  $\Gamma_{AODV}.simps$  [simp del, code del]

**lemmas**  $\Gamma_{AODV}.simps$  [simp, code] =  $\Gamma_{AODV}.simps$  [simplified]

**fun**  $\Gamma_{AODV}.skeleton$

**where**

```

  "ΓAODV.skeleton PAadv = seqp_skeleton (ΓAODV PAadv)"
/ "ΓAODV.skeleton PNewPkt = seqp_skeleton (ΓAODV PNewPkt)"
/ "ΓAODV.skeleton PPkt = seqp_skeleton (ΓAODV PPkt)"
/ "ΓAODV.skeleton PRreq = seqp_skeleton (ΓAODV PRreq)"
/ "ΓAODV.skeleton PRrep = seqp_skeleton (ΓAODV PRrep)"
/ "ΓAODV.skeleton PRerr = seqp_skeleton (ΓAODV PRerr)"

```

**lemma**  $\Gamma_{AODV}.skeleton\_wf$  [simp]:

```

"wellformed ΓAODV.skeleton"
proof (rule, intro allI)

```

```

fix pn pn'
show "call(pn')  $\notin$  stermsl ( $\Gamma_{AODV\_skeleton}$  pn)"
  by (cases pn) simp_all
qed

declare  $\Gamma_{AODV\_skeleton.simps}$  [simp del, code del]
lemmas  $\Gamma_{AODV\_skeleton.simps}$  [simp, code]
  =  $\Gamma_{AODV\_skeleton.simps}$  [simplified  $\Gamma_{AODV\_simps}$  seqp_skeleton.simps]

lemma aodv_proc_cases [dest]:
  fixes p pn
  shows "p  $\in$  ctermsl ( $\Gamma_{AODV}$  pn)  $\implies$ 
    (p  $\in$  ctermsl ( $\Gamma_{AODV}$  PAodv)  $\vee$ 
     p  $\in$  ctermsl ( $\Gamma_{AODV}$  PNewPkt)  $\vee$ 
     p  $\in$  ctermsl ( $\Gamma_{AODV}$  PPkt)  $\vee$ 
     p  $\in$  ctermsl ( $\Gamma_{AODV}$  PRreq)  $\vee$ 
     p  $\in$  ctermsl ( $\Gamma_{AODV}$  PRrep)  $\vee$ 
     p  $\in$  ctermsl ( $\Gamma_{AODV}$  PRerr))"

  using assms
  by (cases pn) simp_all

definition  $\sigma_{AODV} :: "ip \Rightarrow (state \times (state, msg, pseqp, pseqp label) seqp) set"$ 
where " $\sigma_{AODV} i \equiv \{(aodv\_init\ i, \Gamma_{AODV}\ PAodv)\}$ "

abbreviation paodv
  :: "ip  $\Rightarrow$  (state  $\times$  (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
where
  "paodv i  $\equiv$  ( $\{$  init =  $\sigma_{AODV}$  i, trans = seqp_sos  $\Gamma_{AODV}$   $\}$ )"

lemma aodv_trans: "trans (paodv i) = seqp_sos  $\Gamma_{AODV}$ "
  by simp

lemma aodv_control_within [simp]: "control_within  $\Gamma_{AODV}$  (init (paodv i))"
  unfolding  $\sigma_{AODV\_def}$  by (rule control_withinI) (auto simp del:  $\Gamma_{AODV\_simps}$ )

lemma aodv_wf [simp]:
  "wellformed  $\Gamma_{AODV}$ "
  proof (rule, intro allI)
    fix pn pn'
    show "call(pn')  $\notin$  stermsl ( $\Gamma_{AODV}$  pn)"
      by (cases pn) simp_all
  qed

lemmas aodv_labels_not_empty [simp] = labels_not_empty [OF aodv_wf]

lemma aodv_ex_label [intro]: " $\exists l. l \in labels\ \Gamma_{AODV}\ p$ "
  by (metis aodv_labels_not_empty all_not_in_conv)

lemma aodv_ex_labelE [elim]:
  assumes " $\forall l \in labels\ \Gamma_{AODV}\ p. P\ l\ p$ "
  and " $\exists p\ l. P\ l\ p \implies Q$ "
  shows "Q"
  using assms by (metis aodv_ex_label)

lemma aodv_simple_labels [simp]: "simple_labels  $\Gamma_{AODV}$ "
  proof
    fix pn p
    assume "p  $\in$  subterms( $\Gamma_{AODV}$  pn)"
    thus " $\exists l. labels\ \Gamma_{AODV}\ p = \{l\}$ "
      by (cases pn) (simp_all cong: seqp_congs / elim disjE)+
  qed

lemma  $\sigma_{AODV\_labels}$  [simp]: " $(\xi, p) \in \sigma_{AODV}\ i \implies labels\ \Gamma_{AODV}\ p = \{PAodv-:0\}$ "
  unfolding  $\sigma_{AODV\_def}$  by simp

```

```

lemma aadv_init_kD_empty [simp]:
  "(\xi, p) \in \sigma_{AODV} i \implies kD (rt \xi) = \{\}"
  unfolding \sigma_{AODV}_def kD_def by simp

lemma aadv_init_sip_not_ip [simp]: "\neg(sip (aadv_init i) = i)" by simp

lemma aadv_init_sip_not_ip' [simp]:
  assumes "(\xi, p) \in \sigma_{AODV} i"
  shows "sip \xi \neq ip \xi"
  using assms unfolding \sigma_{AODV}_def by simp

lemma aadv_init_sip_not_i [simp]:
  assumes "(\xi, p) \in \sigma_{AODV} i"
  shows "sip \xi \neq i"
  using assms unfolding \sigma_{AODV}_def by simp

lemma clear_locals_sip_not_ip':
  assumes "ip \xi = i"
  shows "\neg(sip (clear_locals \xi) = i)"
  using assms by auto

```

Stop the simplifier from descending into process terms.

```
declare seqp_congs [cong]
```

Configure the main invariant tactic for AODV.

```

declare
  \Gamma_{AODV}_simps [ctermenv]
  aadv_proc_cases [ctermenv_cases]
  seq_invariant_ctermenvI [OF aadv_wf aadv_control_within aadv_simple_labels aadv_trans,
    ctermenv_intros]
  seq_step_invariant_ctermenvI [OF aadv_wf aadv_control_within aadv_simple_labels aadv_trans,
    ctermenv_intros]

end

```

### 3.4 Invariant assumptions and properties

```

theory C_Aadv_Predicates
imports C_Aadv
begin

```

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

```

abbreviation not_Pkt :: "msg \Rightarrow bool"
where "not_Pkt m \equiv case m of Pkt _ _ _ \Rightarrow False | _ \Rightarrow True"

definition msg_sender :: "msg \Rightarrow ip"
where "msg_sender m \equiv case m of Rreq _ _ _ _ _ ipc \Rightarrow ipc
  | Rrep _ _ _ _ ipc \Rightarrow ipc
  | Rerr _ ipc \Rightarrow ipc
  | Pkt _ _ ipc \Rightarrow ipc"

```

```

lemma msg_sender_simps [simp]:
  "\^hops rreqid dip dsn dsk oip osn sip.
    msg_sender (Rreq hops rreqid dip dsn dsk oip osn sip) = sip"
  "\^hops dip dsn oip sip. msg_sender (Rrep hops dip dsn oip sip) = sip"
  "\^dests sip. msg_sender (Rerr dests sip) = sip"
  "\^dip sip. msg_sender (Pkt d dip sip) = sip"
  unfolding msg_sender_def by simp_all

```

```

definition msg_zhops :: "msg \Rightarrow bool"
where "msg_zhops m \equiv case m of
  Rreq hopsc _ dipc _ _ oipc _ sipc \Rightarrow hopsc = 0 \longrightarrow oipc = sipc

```

```

| Rrep hopsd dpc _ _ sipc ⇒ hopsd = 0 → dpc = sipc
| _ ⇒ True"

```

lemma msg\_zhops\_simps [simp]:

```

"∧hops rreqid dip dsn dsk oip osn sip.
  msg_zhops (Rreq hops rreqid dip dsn dsk oip osn sip) = (hops = 0 → oip = sip)"
"∧hops dip dsn oip sip. msg_zhops (Rrep hops dip dsn oip sip) = (hops = 0 → dip = sip)"
"∧dests sip.          msg_zhops (Rerr dests sip)          = True"
"∧d dip.              msg_zhops (Newpkt d dip)            = True"
"∧d dip sip.         msg_zhops (Pkt d dip sip)            = True"
unfolding msg_zhops_def by simp_all

```

definition rreq\_rrep\_sn :: "msg ⇒ bool"

```

where "rreq_rrep_sn m ≡ case m of Rreq _ _ _ _ _ osnc _ ⇒ osnc ≥ 1
| Rrep _ _ dsnc _ _ ⇒ dsnc ≥ 1
| _ ⇒ True"

```

lemma rreq\_rrep\_sn\_simps [simp]:

```

"∧hops rreqid dip dsn dsk oip osn sip.
  rreq_rrep_sn (Rreq hops rreqid dip dsn dsk oip osn sip) = (osn ≥ 1)"
"∧hops dip dsn oip sip. rreq_rrep_sn (Rrep hops dip dsn oip sip) = (dsn ≥ 1)"
"∧dests sip.          rreq_rrep_sn (Rerr dests sip) = True"
"∧d dip.              rreq_rrep_sn (Newpkt d dip) = True"
"∧d dip sip.         rreq_rrep_sn (Pkt d dip sip) = True"
unfolding rreq_rrep_sn_def by simp_all

```

definition rreq\_rrep\_fresh :: "rt ⇒ msg ⇒ bool"

```

where "rreq_rrep_fresh crt m ≡ case m of Rreq hopsd _ _ _ oipc osnc ipcc ⇒ (ipcc ≠ oipc →
  oipc ∈ kD(crt) ∧ (sqn crt oipc > osnc
    ∨ (sqn crt oipc = osnc
      ∧ the (dhops crt oipc) ≤ hopsd
      ∧ the (flag crt oipc) = val)))
| Rrep hopsd dpc dsnc _ ipcc ⇒ (ipcc ≠ dpc →
  dpc ∈ kD(crt)
  ∧ sqn crt dpc = dsnc
  ∧ the (dhops crt dpc) = hopsd
  ∧ the (flag crt dpc) = val)
| _ ⇒ True"

```

lemma rreq\_rrep\_fresh\_simps [simp]:

```

"∧hops rreqid dip dsn dsk oip osn sip.
  rreq_rrep_fresh crt (Rreq hops rreqid dip dsn dsk oip osn sip) =
  (sip ≠ oip → oip ∈ kD(crt)
  ∧ (sqn crt oip > osn
    ∨ (sqn crt oip = osn
      ∧ the (dhops crt oip) ≤ hops
      ∧ the (flag crt oip) = val))))"
"∧hops dip dsn oip sip. rreq_rrep_fresh crt (Rrep hops dip dsn oip sip) =
  (sip ≠ dip → dip ∈ kD(crt)
  ∧ sqn crt dip = dsn
  ∧ the (dhops crt dip) = hops
  ∧ the (flag crt dip) = val)"
"∧dests sip.          rreq_rrep_fresh crt (Rerr dests sip) = True"
"∧d dip.              rreq_rrep_fresh crt (Newpkt d dip) = True"
"∧d dip sip.         rreq_rrep_fresh crt (Pkt d dip sip) = True"
unfolding rreq_rrep_fresh_def by simp_all

```

definition rerr\_invalid :: "rt ⇒ msg ⇒ bool"

```

where "rerr_invalid crt m ≡ case m of Rerr destsc _ ⇒ (∃ripc ∈ dom(destsc).
  (ripc ∈ iD(crt) ∧ the (destsc ripc) = sqn crt ripc))
| _ ⇒ True"

```

lemma rerr\_invalid\_simps [simp]:

```

"∧hops rreqid dip dsn dsk oip osn sip.

```

```

      rerr_invalid crt (Rreq hops rreqid dip dsn dsk oip osn sip) = True"
"∧hops dip dsn oip sip. rerr_invalid crt (Rrep hops dip dsn oip sip) = True"
"∧dests sip. rerr_invalid crt (Rerr dests sip) = (∀rip∈dom(dests).
      rip∈iD(crt) ∧ the (dests rip) = sqn crt rip)"
"∧d dip. rerr_invalid crt (Newpkt d dip) = True"
"∧d dip sip. rerr_invalid crt (Pkt d dip sip) = True"
unfolding rerr_invalid_def by simp_all

```

#### definition

```
initmissing :: "(nat ⇒ state option) × 'a ⇒ (nat ⇒ state) × 'a"
```

#### where

```
"initmissing σ = (λi. case (fst σ) i of None ⇒ aadv_init i | Some s ⇒ s, snd σ)"
```

```
lemma not_in_net_ips_fst_init_missing [simp]:
```

```
assumes "i ∉ net_ips σ"
```

```
shows "fst (initmissing (netgmap fst σ)) i = aadv_init i"
```

```
using assms unfolding initmissing_def by simp
```

```
lemma fst_initmissing_netgmap_pair_fst [simp]:
```

```
"fst (initmissing (netgmap (λ(p, q). (fst (id p), snd (id p), q)) s))
```

```
= fst (initmissing (netgmap fst s))"
```

```
unfolding initmissing_def by auto
```

We introduce a streamlined alternative to `initmissing` with `netgmap` to simplify invariant statements and thus facilitate their comprehension and presentation.

```
lemma fst_initmissing_netgmap_default_aadv_init_netlift:
```

```
"fst (initmissing (netgmap fst s)) = default aadv_init (netlift fst s)"
```

```
unfolding initmissing_def default_def
```

```
by (simp add: fst_netgmap_netlift del: One_nat_def)
```

#### definition

```
netglobal :: "((nat ⇒ state) ⇒ bool) ⇒ ((state × 'b) × 'c) net_state ⇒ bool"
```

#### where

```
"netglobal P ≡ (λs. P (default aadv_init (netlift fst s)))"
```

end

## 3.5 Quality relations between routes

```
theory C_Fresher
```

```
imports C_Aadv_Data
```

```
begin
```

### 3.5.1 Net sequence numbers

#### On individual routes

#### definition

```
nsqnr :: "r ⇒ sqn"
```

#### where

```
"nsqnr r ≡ if π4(r) = val ∨ π2(r) = 0 then π2(r) else (π2(r) - 1)"
```

```
lemma nsqnr_def':
```

```
"nsqnr r = (if π4(r) = inv then π2(r) - 1 else π2(r))"
```

```
unfolding nsqnr_def by simp
```

```
lemma nsqnr_zero [simp]:
```

```
"∧dsn dsk flag hops nhip. nsqnr (0, dsk, flag, hops, nhip) = 0"
```

```
unfolding nsqnr_def by clarsimp
```

```
lemma nsqnr_val [simp]:
```

```
"∧dsn dsk hops nhip. nsqnr (dsn, dsk, val, hops, nhip) = dsn"
```

```
unfolding nsqnr_def by clarsimp
```



```
lemma nsqn_r_inv [simp]:
  " $\bigwedge$  dsn dsk hops nhip. nsqn_r (dsn, dsk, inv, hops, nhip) = dsn - 1"
  unfolding nsqn_r_def by clarsimp
```

```
lemma nsqn_r_lte_dsn [simp]:
  " $\bigwedge$  dsn dsk flag hops nhip. nsqn_r (dsn, dsk, flag, hops, nhip)  $\leq$  dsn"
  unfolding nsqn_r_def by clarsimp
```

## On routes in routing tables

### definition

```
nsqn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  sqn"
where
  "nsqn  $\equiv$   $\lambda$ rt dip. case  $\sigma_{route}$ (rt, dip) of None  $\Rightarrow$  0 | Some r  $\Rightarrow$  nsqn_r(r)"
```

```
lemma nsqn_sqn_def:
  " $\bigwedge$ rt dip. nsqn rt dip = (if flag rt dip = Some val  $\vee$  sqn rt dip = 0
    then sqn rt dip else sqn rt dip - 1)"
  unfolding nsqn_def sqn_def by (clarsimp split: option.split) auto
```

```
lemma not_in_kD_nsqn [simp]:
  assumes "dip  $\notin$  kD(rt)"
  shows "nsqn rt dip = 0"
  using assms unfolding nsqn_def by simp
```

```
lemma kD_nsqn:
  assumes "dip  $\in$  kD(rt)"
  shows "nsqn rt dip = nsqn_r(the ( $\sigma_{route}$ (rt, dip)))"
  using assms [THEN kD_Some] unfolding nsqn_def by clarsimp
```

```
lemma nsqn_r_flag_pred [simp, intro]:
  fixes dsn dsk flag hops nhip
  assumes "P (nsqn_r (dsn, dsk, val, hops, nhip))"
  and "P (nsqn_r (dsn, dsk, inv, hops, nhip))"
  shows "P (nsqn_r (dsn, dsk, flag, hops, nhip))"
  using assms by (cases flag) auto
```

```
lemma sqn_nsqn:
  " $\bigwedge$ rt dip. sqn rt dip - 1  $\leq$  nsqn rt dip"
  unfolding sqn_def nsqn_def by (clarsimp split: option.split)
```

```
lemma nsqn_sqn: "nsqn rt dip  $\leq$  sqn rt dip"
  unfolding sqn_def nsqn_def by (cases "rt dip") auto
```

```
lemma val_nsqn_sqn [elim, simp]:
  assumes "ip  $\in$  kD(rt)"
  and "the (flag rt ip) = val"
  shows "nsqn rt ip = sqn rt ip"
  using assms unfolding nsqn_sqn_def by auto
```

```
lemma vD_nsqn_sqn [elim, simp]:
  assumes "ip  $\in$  vD(rt)"
  shows "nsqn rt ip = sqn rt ip"
  proof -
    from (ip  $\in$  vD(rt)) have "ip  $\in$  kD(rt)"
    and "the (flag rt ip) = val" by auto
  thus ?thesis ..
qed
```

```
lemma inv_nsqn_sqn [elim, simp]:
  assumes "ip  $\in$  kD(rt)"
  and "the (flag rt ip) = inv"
  shows "nsqn rt ip = sqn rt ip - 1"
  using assms unfolding nsqn_sqn_def by auto
```

```

lemma iD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ iD(rt)"
  shows "nsqn rt ip = sqn rt ip - 1"
proof -
  from ⟨ip ∈ iD(rt)⟩ have "ip ∈ kD(rt)"
    and "the (flag rt ip) = inv" by auto
  thus ?thesis ..
qed

lemma nsqn_update_changed_kno_val [simp]: "\rt ip dsn dsk hops nhip.
rt ≠ update rt ip (dsn, kno, val, hops, nhip)
⇒ nsqn (update rt ip (dsn, kno, val, hops, nhip)) ip = dsn"
using assms unfolding nsqn_r_def update_def
by (clarsimp simp: kD_nsqn split: option.split_asm option.split_if_asm)
(metis fun_upd_triv)

lemma nsqn_update_other [simp]:
  fixes dsn dsk flag hops dip nhip rt ip
  assumes "dip ≠ ip"
  shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip)) dip = nsqn rt dip"
using assms unfolding nsqn_def
by (clarsimp split: option.split)

lemma nsqn_invalidate_eq:
  assumes "dip ∈ kD(rt)"
  and "dests dip = Some rsn"
  shows "nsqn (invalidate rt dests) dip = rsn - 1"
using assms
proof -
  from assms obtain dsk hops nhip pre
  where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip)"
  unfolding invalidate_def
  by auto
  moreover from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp
  ultimately show ?thesis
  using ⟨dests dip = Some rsn⟩ by simp
qed

lemma nsqn_invalidate_other [simp]:
  assumes "dip ∈ kD(rt)"
  and "dip ∉ dom dests"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
using assms by (clarsimp simp add: kD_nsqn)

```

### 3.5.2 Comparing routes

definition

```
fresher :: "r ⇒ r ⇒ bool" ("(_/ ⊆ _)" [51, 51] 50)
```

where

```
"fresher r r' ≡ ((nsqn_r r < nsqn_r r') ∨ (nsqn_r r = nsqn_r r' ∧ π5(r) ≥ π5(r')))"
```

lemma fresherI1 [intro]:

```
assumes "nsqn_r r < nsqn_r r'"
shows "r ⊆ r'"
unfolding fresher_def using assms by simp
```

lemma fresherI2 [intro]:

```
assumes "nsqn_r r = nsqn_r r'"
and "π5(r) ≥ π5(r')"
shows "r ⊆ r'"
unfolding fresher_def using assms by simp
```

lemma fresherI [intro]:

```

assumes "(nsqnr r < nsqnr r') ∨ (nsqnr r = nsqnr r' ∧ π5(r) ≥ π5(r'))"
shows "r ⊆ r'"
unfolding fresher_def using assms .

```

```

lemma fresherE [elim]:
  assumes "r ⊆ r'"
    and "nsqnr r < nsqnr r' ⇒ P r r'"
    and "nsqnr r = nsqnr r' ∧ π5(r) ≥ π5(r') ⇒ P r r'"
  shows "P r r'"
using assms unfolding fresher_def by auto

```

```

lemma fresher_refl [simp]: "r ⊆ r"
unfolding fresher_def by simp

```

```

lemma fresher_trans [elim, trans]:
  "[[ x ⊆ y; y ⊆ z ] ⇒ x ⊆ z"
unfolding fresher_def by auto

```

```

lemma not_fresher_trans [elim, trans]:
  "[[ ¬(x ⊆ y); ¬(z ⊆ x) ] ⇒ ¬(z ⊆ y)"
unfolding fresher_def by auto

```

```

lemma fresher_dsn_flag_hops_const [simp]:
  fixes dsn dsk' flag hops nhip'
  shows "(dsn, dsk, flag, hops, nhip) ⊆ (dsn, dsk', flag, hops, nhip)"
unfolding fresher_def by (cases flag) simp_all

```

### 3.5.3 Comparing routing tables

definition

```

rt_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"

```

where

```

"rt_fresher ≡ λdip rt rt'. (the (σroute(rt, dip))) ⊆ (the (σroute(rt', dip)))"

```

abbreviation

```

rt_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊆_ _)" [51, 999, 51] 50)

```

where

```

"rt1 ⊆i rt2 ≡ rt_fresher i rt1 rt2"

```

lemma rt\_fresher\_def':

```

"(rt1 ⊆i rt2) = (nsqnr (the (rt1 i)) < nsqnr (the (rt2 i)) ∨
  nsqnr (the (rt1 i)) = nsqnr (the (rt2 i)) ∧ π5 (the (rt2 i)) ≤ π5 (the (rt1 i)))"
unfolding rt_fresher_def fresher_def by (rule refl)

```

lemma single\_rt\_fresher [intro]:

```

assumes "the (rt1 ip) ⊆ the (rt2 ip)"
shows "rt1 ⊆ip rt2"
using assms unfolding rt_fresher_def .

```

lemma rt\_fresher\_single [intro]:

```

assumes "rt1 ⊆ip rt2"
shows "the (rt1 ip) ⊆ the (rt2 ip)"
using assms unfolding rt_fresher_def .

```

lemma rt\_fresher\_def2:

```

assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
shows "(rt1 ⊆dip rt2) = (nsqn rt1 dip < nsqn rt2 dip
  ∨ (nsqn rt1 dip = nsqn rt2 dip
  ∧ the (dhops rt1 dip) ≥ the (dhops rt2 dip)))"
using assms unfolding rt_fresher_def fresher_def by (simp add: kD_nsqn proj5_eq_dhops)

```

lemma rt\_fresherI1 [intro]:

```

assumes "dip ∈ kD(rt1)"

```

```

    and "dip ∈ kD(rt2)"
    and "nsqn rt1 dip < nsqn rt2 dip"
  shows "rt1 ⊆dip rt2"
  unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt_fresherI2 [intro]:
  assumes "dip ∈ kD(rt1)"
    and "dip ∈ kD(rt2)"
    and "nsqn rt1 dip = nsqn rt2 dip"
    and "the (dhops rt1 dip) ≥ the (dhops rt2 dip)"
  shows "rt1 ⊆dip rt2"
  unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3-4) by simp

lemma rt_fresherE [elim]:
  assumes "rt1 ⊆dip rt2"
    and "dip ∈ kD(rt1)"
    and "dip ∈ kD(rt2)"
    and "[| nsqn rt1 dip < nsqn rt2 dip |] ⇒ P rt1 rt2 dip"
    and "[| nsqn rt1 dip = nsqn rt2 dip;
      the (dhops rt1 dip) ≥ the (dhops rt2 dip) |] ⇒ P rt1 rt2 dip"
  shows "P rt1 rt2 dip"
  using assms(1) unfolding rt_fresher_def2 [OF assms(2-3)]
  using assms(4-5) by auto

lemma rt_fresher_refl [simp]: "rt ⊆dip rt"
  unfolding rt_fresher_def by simp

lemma rt_fresher_trans [elim, trans]:
  assumes "rt1 ⊆dip rt2"
    and "rt2 ⊆dip rt3"
  shows "rt1 ⊆dip rt3"
  using assms unfolding rt_fresher_def by auto

lemma rt_fresher_if_Some [intro!]:
  assumes "the (rt dip) ⊆ r"
  shows "rt ⊆dip (λip. if ip = dip then Some r else rt ip)"
  using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip ⇒ rt ⇒ rt ⇒ bool"
where
  "rt_fresh_as ≡ λdip rt1 rt2. (rt1 ⊆dip rt2) ∧ (rt2 ⊆dip rt1)"

abbreviation
  rt_fresh_as_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ≈  _)" [51, 999, 51] 50)
where
  "rt1 ≈i rt2 ≡ rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_refl [simp]: "∧rt dip. rt ≈dip rt"
  unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
  "∧rt1 rt2 rt3 dip. [| rt1 ≈dip rt2; rt2 ≈dip rt3 |] ⇒ rt1 ≈dip rt3"
  unfolding rt_fresh_as_def rt_fresher_def
  by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
  assumes "rt1 ⊆dip rt2"
    and "rt2 ⊆dip rt1"
  shows "rt1 ≈dip rt2"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
  assumes "dip ∈ kD(rt1)"
    and "dip ∈ kD(rt2)"

```

```

    and "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"
    and "the (rt2 dip)  $\sqsubseteq$  the (rt1 dip)"
  shows "rt1  $\approx_{dip}$  rt2"
using assms unfolding rt_fresh_as_def
by (clarsimp dest!: single_rt_fresher)

lemma nsqn_rt_fresh_asI:
  assumes "dip  $\in$  kD(rt)"
    and "dip  $\in$  kD(rt')"
    and "nsqn rt dip = nsqn rt' dip"
    and " $\pi_5(\text{the } (rt \text{ dip})) = \pi_5(\text{the } (rt' \text{ dip}))$ "
  shows "rt  $\approx_{dip}$  rt'"
proof
  from assms(1-2,4) have dhops': "the (dhops rt' dip)  $\leq$  the (dhops rt dip)"
  by (simp add: proj5_eq_dhops)
  with assms(1-3) show "rt  $\sqsubseteq_{dip}$  rt'"
  by (rule rt_fresherI2)
next
  from assms(1-2,4) have dhops: "the (dhops rt dip)  $\leq$  the (dhops rt' dip)"
  by (simp add: proj5_eq_dhops)
  with assms(2,1) assms(3) [symmetric] show "rt'  $\sqsubseteq_{dip}$  rt"
  by (rule rt_fresherI2)
qed

lemma rt_fresh_asE [elim]:
  assumes "rt1  $\approx_{dip}$  rt2"
    and "[| rt1  $\sqsubseteq_{dip}$  rt2; rt2  $\sqsubseteq_{dip}$  rt1 |]  $\implies$  P rt1 rt2 dip"
  shows "P rt1 rt2 dip"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD1 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\sqsubseteq_{dip}$  rt1"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\approx_{dip}$  rt1"
using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
  assumes " $\neg$  (rt1  $\sqsubseteq_{dip}$  rt2)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
proof
  assume "rt1  $\approx_{dip}$  rt2"
  hence "rt1  $\sqsubseteq_{dip}$  rt2" ..
  with ( $\neg$  (rt1  $\sqsubseteq_{dip}$  rt2)) show False ..
qed

lemma not_rt_fresh_asI2 [intro]:
  assumes " $\neg$  (rt2  $\sqsubseteq_{dip}$  rt1)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
proof
  assume "rt1  $\approx_{dip}$  rt2"
  hence "rt2  $\sqsubseteq_{dip}$  rt1" ..
  with ( $\neg$  (rt2  $\sqsubseteq_{dip}$  rt1)) show False ..
qed

lemma not_single_rt_fresher [elim]:

```

```

assumes "¬(the (rt1 ip) ⊆ the (rt2 ip))"
shows "¬(rt1 ⊆ip rt2)"
proof
  assume "rt1 ⊆ip rt2"
  hence "the (rt1 ip) ⊆ the (rt2 ip)" ..
  with (¬(the (rt1 ip) ⊆ the (rt2 ip))) show False ..
qed

lemmas not_single_rt_fresh_asI1 [intro] = not_rt_fresh_asI1 [OF not_single_rt_fresher]
lemmas not_single_rt_fresh_asI2 [intro] = not_rt_fresh_asI2 [OF not_single_rt_fresher]

lemma not_rt_fresher_single [elim]:
  assumes "¬(rt1 ⊆ip rt2)"
  shows "¬(the (rt1 ip) ⊆ the (rt2 ip))"
proof
  assume "the (rt1 ip) ⊆ the (rt2 ip)"
  hence "rt1 ⊆ip rt2" ..
  with (¬(rt1 ⊆ip rt2)) show False ..
qed

lemma rt_fresh_as_nsqnr:
  assumes "dip ∈ kD(rt1)"
    and "dip ∈ kD(rt2)"
    and "rt1 ≈dip rt2"
  shows "nsqnr (the (rt2 dip)) = nsqnr (the (rt1 dip))"
using assms(3) unfolding rt_fresh_as_def
by (auto simp: rt_fresher_def2 [OF ⟨dip ∈ kD(rt1)⟩ ⟨dip ∈ kD(rt2)⟩]
    rt_fresher_def2 [OF ⟨dip ∈ kD(rt2)⟩ ⟨dip ∈ kD(rt1)⟩]
    kD_nsqn [OF ⟨dip ∈ kD(rt1)⟩]
    kD_nsqn [OF ⟨dip ∈ kD(rt2)⟩])

lemma rt_fresher_mapupd [intro!]:
  assumes "dip ∈ kD(rt)"
    and "the (rt dip) ⊆ r"
  shows "rt ⊆dip rt(dip ↦ r)"
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_map_update_other [intro!]:
  assumes "dip ∈ kD(rt)"
    and "dip ≠ ip"
  shows "rt ⊆dip rt(ip ↦ r)"
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_update_other [simp]:
  assumes inkD: "dip ∈ kD(rt)"
    and "dip ≠ ip"
  shows "rt ⊆dip update rt ip r"
using assms unfolding update_def
by (clarsimp split: option.split) (fastforce)

theorem rt_fresher_update [simp]:
  assumes "dip ∈ kD(rt)"
    and "the (dhops rt dip) ≥ 1"
    and "update_arg_wf r"
  shows "rt ⊆dip update rt ip r"
proof (cases "dip = ip")
  assume "dip ≠ ip" with ⟨dip ∈ kD(rt)⟩ show ?thesis
  by (rule rt_fresher_update_other)
next
  assume "dip = ip"

  from ⟨dip ∈ kD(rt)⟩ obtain dsnn dskn fn hopsn nhipn
  where rtn [simp]: "the (rt dip) = (dsnn, dskn, fn, hopsn, nhipn)"
  by (metis prod_cases5)

```

```

with ⟨the ⟨dhops rt dip⟩ ≥ 1⟩ and ⟨dip ∈ kD(rt)⟩ have "hopsn ≥ 1"
  by ⟨metis proj5_eq_dhops projs(4)⟩
from ⟨dip ∈ kD(rt)⟩ rtn have [simp]: "sqn rt dip = dsnn"
  and [simp]: "the ⟨dhops rt dip⟩ = hopsn"
  and [simp]: "the ⟨flag rt dip⟩ = fn"
  by (simp add: sqn_def proj5_eq_dhops [symmetric]
      proj4_eq_flag [symmetric])
from ⟨update_arg_wf r⟩ have "(dsnn, dskn, fn, hopsn, nhipn)
  ⊆ the ⟨⟨update rt dip r⟩ dip⟩"
proof (rule wf_r_cases)
  fix nhip pre
  from ⟨hopsn ≥ 1⟩ have "∧pre'. (dsnn, dskn, fn, hopsn, nhipn)
    ⊆ (dsnn, unk, val, Suc 0, nhip)"
  unfolding fresher_def sqn_def by (cases fn) auto
  thus "(dsnn, dskn, fn, hopsn, nhipn)
    ⊆ the ⟨⟨update rt dip (0, unk, val, Suc 0, nhip) dip⟩⟩"
  using ⟨dip ∈ kD(rt)⟩ by - (rule update_cases_kD, simp_all)
next
fix dsn :: sqn and hops nhip pre
assume "0 < dsn"
show "(dsnn, dskn, fn, hopsn, nhipn)
  ⊆ the ⟨⟨update rt dip (dsn, kno, val, hops, nhip) dip⟩⟩"
proof (rule update_cases_kD [OF _ ⟨dip ∈ kD(rt)⟩], simp_all add: ⟨0 < dsn⟩)
  assume "dsnn < dsn"
  thus "(dsnn, dskn, fn, hopsn, nhipn)
    ⊆ (dsn, kno, val, hops, nhip)"
  unfolding fresher_def by auto
next
  assume "dsnn = dsn"
  and "hops < hopsn"
  thus "(dsn, dskn, fn, hopsn, nhipn)
    ⊆ (dsn, kno, val, hops, nhip)"
  unfolding fresher_def nsqn_r_def by simp
next
  assume "dsnn = dsn"
  with ⟨0 < dsn⟩
  show "(dsn, dskn, inv, hopsn, nhipn)
    ⊆ (dsn, kno, val, hops, nhip)"
  unfolding fresher_def by simp
qed
qed
hence "rt ⊆dip update rt dip r"
  by - (rule single_rt_fresher, simp)
with ⟨dip = ip⟩ show ?thesis by simp
qed

```

```

theorem rt_fresher_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
  and indests: "∀rip ∈ dom(dests). rip ∈ vD(rt) ∧ sqn rt rip < the (dests rip)"
  shows "rt ⊆dip invalidate rt dests"
proof (cases "dip ∈ dom(dests)")
  assume "dip ∉ dom(dests)"
  thus ?thesis using ⟨dip ∈ kD(rt)⟩
  by - (rule single_rt_fresher, simp)
next
  assume "dip ∈ dom(dests)"
  moreover with indests have "dip ∈ vD(rt)"
  and "sqn rt dip < the (dests dip)"
  by auto
  ultimately show ?thesis
  unfolding invalidate_def sqn_def
  by - (rule single_rt_fresher, auto simp: fresher_def)
qed

```

```

lemma nsqnr_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
    and "dip ∈ dom(dests)"
  shows "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
  using assms unfolding invalidate_def by auto

lemma rt_fresh_as_inc_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
    and "∀ rip ∈ dom(dests). rip ∈ vD(rt) ∧ the (dests rip) = inc (sqn rt rip)"
  shows "rt ≈dip invalidate rt dests"
proof (cases "dip ∈ dom(dests)")
  assume "dip ∉ dom(dests)"
  with ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)"
    by simp
  with ⟨dip ∈ kD(rt)⟩ show ?thesis
    by rule (simp_all add: ⟨dip ∉ dom(dests)⟩)
next
  assume "dip ∈ dom(dests)"
  with assms(2) have "dip ∈ vD(rt)"
    and "the (dests dip) = inc (sqn rt dip)" by auto
  from ⟨dip ∈ vD(rt)⟩ have "dip ∈ kD(rt)" by simp
  moreover then have "dip ∈ kD(invalidate rt dests)" by simp
  ultimately show ?thesis
proof (rule nsqnr_rt_fresh_asI)
  from ⟨dip ∈ vD(rt)⟩ have "nsqn rt dip = sqn rt dip" by simp
  also have "sqn rt dip = nsqnr (the (invalidate rt dests dip))"
  proof -
    from ⟨dip ∈ kD(rt)⟩ have "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
      using ⟨dip ∈ dom(dests)⟩ by (rule nsqnr_invalidate)
    with ⟨the (dests dip) = inc (sqn rt dip)⟩
      show "sqn rt dip = nsqnr (the (invalidate rt dests dip))" by simp
  qed
  also from ⟨dip ∈ kD(invalidate rt dests)⟩
    have "nsqnr (the (invalidate rt dests dip)) = nsqn (invalidate rt dests) dip"
      by (simp add: kD_nsqn)
  finally show "nsqn rt dip = nsqn (invalidate rt dests) dip" .
qed simp
qed

```

lemmas `rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]`

### 3.5.4 Strictly comparing routing tables

definition `rt_strictly_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"`

where

`"rt_strictly_fresher ≡ λdip rt1 rt2. (rt1 ⊆dip rt2) ∧ ¬(rt1 ≈dip rt2)"`

abbreviation

`rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊆_ _)" [51, 999, 51] 50)`

where

`"rt1 ⊆i rt2 ≡ rt_strictly_fresher i rt1 rt2"`

lemma `rt_strictly_fresher_def''`:

`"rt1 ⊆i rt2 = ((rt1 ⊆i rt2) ∧ ¬(rt2 ⊆i rt1))"`

unfolding `rt_strictly_fresher_def` `rt_fresh_as_def` by auto

lemma `rt_strictly_fresherI'` [intro]:

assumes `"rt1 ⊆i rt2"`

and `"¬(rt2 ⊆i rt1)"`

shows `"rt1 ⊆i rt2"`

using `assms` unfolding `rt_strictly_fresher_def''` by `simp`

lemma `rt_strictly_fresherE'` [elim]:



```

assumes "rt1  $\sqsubseteq_i$  rt2"
  and "[[ rt1  $\sqsubseteq_i$  rt2;  $\neg$ (rt2  $\sqsubseteq_i$  rt1) ]  $\implies$  P rt1 rt2 i"
  shows "P rt1 rt2 i"
using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherI [intro]:
  assumes "rt1  $\sqsubseteq_i$  rt2"
  and " $\neg$ (rt1  $\approx_i$  rt2)"
  shows "rt1  $\sqsubseteq_i$  rt2"
unfolding rt_strictly_fresher_def using assms ..

lemmas rt_strictly_fresher_singleI [elim] = rt_strictly_fresherI [OF single_rt_fresher]

lemma rt_strictly_fresherE [elim]:
  assumes "rt1  $\sqsubseteq_i$  rt2"
  and "[[ rt1  $\sqsubseteq_i$  rt2;  $\neg$ (rt1  $\approx_i$  rt2) ]  $\implies$  P rt1 rt2 i"
  shows "P rt1 rt2 i"
using assms(1) unfolding rt_strictly_fresher_def
by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':
  "rt1  $\sqsubseteq_i$  rt2 =
  (nsqnr (the (rt1 i)) < nsqnr (the (rt2 i))
   $\vee$  (nsqnr (the (rt1 i)) = nsqnr (the (rt2 i))  $\wedge$   $\pi_5$ (the (rt1 i)) >  $\pi_5$ (the (rt2 i))))"
unfolding rt_strictly_fresher_def'' rt_fresher_def fresher_def by auto

lemma rt_strictly_fresher_fresherD [dest]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  shows "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"
using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  shows " $\neg$  rt1  $\approx_{dip}$  rt2"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
using assms proof -
  from (rt1  $\sqsubseteq_{dip}$  rt2) obtain "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)" by auto
  also from (rt2  $\sqsubseteq_{dip}$  rt3) obtain "the (rt2 dip)  $\sqsubseteq$  the (rt3 dip)" by auto
  finally have "the (rt1 dip)  $\sqsubseteq$  the (rt3 dip)" .

  moreover have " $\neg$  (rt1  $\approx_{dip}$  rt3)"
  proof -
    from (rt1  $\sqsubseteq_{dip}$  rt2) obtain " $\neg$ (the (rt2 dip)  $\sqsubseteq$  the (rt1 dip))" by auto
    also from (rt2  $\sqsubseteq_{dip}$  rt3) obtain " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt2 dip))" by auto
    finally have " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt1 dip))" .
    thus ?thesis ..
  qed
  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: " $\neg$  (rt  $\sqsubseteq_{dip}$  rt)"
using assms unfolding rt_strictly_fresher_def
by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
proof -

```

```

from ⟨rt1 ⊆dip rt2⟩ have "rt1 ⊆dip rt2"
      and "¬(rt2 ⊆dip rt1)"
  unfolding rt_strictly_fresher_def'' by auto
from this(1) and ⟨rt2 ⊆dip rt3⟩ have "rt1 ⊆dip rt3" ..

moreover from ⟨¬(rt2 ⊆dip rt1)⟩ have "¬(rt3 ⊆dip rt1)"
  proof (rule contrapos_nn)
    assume "rt3 ⊆dip rt1"
    with ⟨rt2 ⊆dip rt3⟩ show "rt2 ⊆dip rt1" ..
  qed

ultimately show "rt1 ⊆dip rt3"
  unfolding rt_strictly_fresher_def'' by auto
qed

lemma rt_fresher_trans_rt_strictly_fresher' [elim, trans]:
  assumes "rt1 ⊆dip rt2"
    and "rt2 ⊆dip rt3"
  shows "rt1 ⊆dip rt3"
  proof -
    from ⟨rt2 ⊆dip rt3⟩ have "rt2 ⊆dip rt3"
      and "¬(rt3 ⊆dip rt2)"
    unfolding rt_strictly_fresher_def'' by auto
    from ⟨rt1 ⊆dip rt2⟩ and this(1) have "rt1 ⊆dip rt3" ..

    moreover from ⟨¬(rt3 ⊆dip rt2)⟩ have "¬(rt3 ⊆dip rt1)"
    proof (rule contrapos_nn)
      assume "rt3 ⊆dip rt1"
      thus "rt3 ⊆dip rt2" using ⟨rt1 ⊆dip rt2⟩ ..
    qed

    ultimately show "rt1 ⊆dip rt3"
      unfolding rt_strictly_fresher_def'' by auto
  qed

lemma rt_fresher_imp_nsqn_le:
  assumes "rt1 ⊆ip rt2"
    and "ip ∈ kD rt1"
    and "ip ∈ kD rt2"
  shows "nsqn rt1 ip ≤ nsqn rt2 ip"
  using assms(1)
  by (auto simp add: rt_fresher_def2 [OF assms(2-3)])

lemma rt_strictly_fresher_ltI [intro]:
  assumes "dip ∈ kD(rt1)"
    and "dip ∈ kD(rt2)"
    and "nsqn rt1 dip < nsqn rt2 dip"
  shows "rt1 ⊆dip rt2"
  proof
    from assms show "rt1 ⊆dip rt2" ..
  next
    show "¬(rt1 ≈dip rt2)"
    proof
      assume "rt1 ≈dip rt2"
      hence "rt2 ⊆dip rt1" ..
      hence "nsqn rt2 dip ≤ nsqn rt1 dip"
        using ⟨dip ∈ kD(rt2)⟩ ⟨dip ∈ kD(rt1)⟩
        by (rule rt_fresher_imp_nsqn_le)
      with ⟨nsqn rt1 dip < nsqn rt2 dip⟩ show "False"
        by simp
    qed
  qed

lemma rt_strictly_fresher_eqI [intro]:

```

```

assumes "i ∈ kD(rt1)"
  and "i ∈ kD(rt2)"
  and "nsqn rt1 i = nsqn rt2 i"
  and "π5(the (rt2 i)) < π5(the (rt1 i))"
shows "rt1 ⊑i rt2"
using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

```

```

lemma invalidate_rtsf_left [simp]:
  "∧dests dip rt rt'. dests dip = None ⇒ (invalidate rt dests ⊑dip rt') = (rt ⊑dip rt')"
unfolding invalidate_def rt_strictly_fresher_def'
by (rule iffI) (auto split: option.split_asm)

```

```

lemma vD_invalidate_rt_strictly_fresher [simp]:
  assumes "dip ∈ vD(invalidate rt1 dests)"
  shows "(invalidate rt1 dests ⊑dip rt2) = (rt1 ⊑dip rt2)"
proof (cases "dip ∈ dom(dests)")
  assume "dip ∈ dom(dests)"
  hence "dip ∉ vD(invalidate rt1 dests)"
  unfolding invalidate_def vD_def
  by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
  with ⟨dip ∈ vD(invalidate rt1 dests)⟩ show ?thesis by simp
next
  assume "dip ∉ dom(dests)"
  hence "dests dip = None" by auto
  moreover with ⟨dip ∈ vD(invalidate rt1 dests)⟩ have "dip ∈ vD(rt1)"
  unfolding invalidate_def vD_def
  by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
  ultimately show ?thesis
  unfolding invalidate_def rt_strictly_fresher_def' by auto
qed

```

```

lemma rt_strictly_fresher_update_other [elim!]:
  "∧dip ip rt r rt'. [ dip ≠ ip; rt ⊑dip rt' ] ⇒ update rt ip r ⊑dip rt'"
unfolding rt_strictly_fresher_def' by clarsimp

```

```

lemma lt_sqn_imp_update_strictly_fresher:
  assumes "dip ∈ vD (rt2 nhip)"
  and *: "osn < sqn (rt2 nhip) dip"
  and **: "rt ≠ update rt dip (osn, kno, val, hops, nhip)"
  shows "update rt dip (osn, kno, val, hops, nhip) ⊑dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI1)
  from ** have "nsqn (update rt dip (osn, kno, val, hops, nhip)) dip = osn"
  by (rule nsqn_update_changed_kno_val)
  with ⟨dip ∈ vD(rt2 nhip)⟩
  have "nsqnr (the (update rt dip (osn, kno, val, hops, nhip) dip)) = osn"
  by (simp add: kD_nsqn)
  also have "osn < sqn (rt2 nhip) dip" by (rule *)
  also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip dip))"
  unfolding nsqnr_def using ⟨dip ∈ vD (rt2 nhip)⟩
  by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show "nsqnr (the (update rt dip (osn, kno, val, hops, nhip) dip))
    < nsqnr (the (rt2 nhip dip))" .
qed

```

```

lemma dhops_le_hops_imp_update_strictly_fresher:
  assumes "dip ∈ vD(rt2 nhip)"
  and sqn: "sqn (rt2 nhip) dip = osn"
  and hop: "the (dhops (rt2 nhip) dip) ≤ hops"
  and **: "rt ≠ update rt dip (osn, kno, val, Suc hops, nhip)"
  shows "update rt dip (osn, kno, val, Suc hops, nhip) ⊑dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI2, rule conjI)
  from ** have "nsqn (update rt dip (osn, kno, val, Suc hops, nhip)) dip = osn"

```

```

  by (rule nsqn_update_changed_kno_val)
with ⟨dip ∈ vD(rt2 nhip)⟩
  have "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip) dip)) = osn"
  by (simp add: kD_nsqn)
also have "osn = sqn (rt2 nhip) dip" by (rule sqn [symmetric])
also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip) dip)"
  unfolding nsqnr_def using ⟨dip ∈ vD(rt2 nhip)⟩
  by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
finally show "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip) dip))
              = nsqnr (the (rt2 nhip) dip))" .

next
  have "the (dhops (rt2 nhip) dip) ≤ hops" by (rule hop)
  also have "hops < hops + 1" by simp
  also have "hops + 1 = the (dhops (update rt dip (osn, kno, val, Suc hops, nhip)) dip)"
    using ** by simp
  finally have "the (dhops (rt2 nhip) dip)
               < the (dhops (update rt dip (osn, kno, val, Suc hops, nhip)) dip)" .
  thus "π5 (the (rt2 nhip) dip) < π5 (the (update rt dip (osn, kno, val, Suc hops, nhip) dip))"
    using ⟨dip ∈ vD(rt2 nhip)⟩ by (simp add: proj5_eq_dhops)
qed

lemma nsqn_invalidate:
  assumes "dip ∈ kD(rt)"
    and "∀ ip ∈ dom(dests). ip ∈ vD(rt) ∧ the (dests ip) = inc (sqn rt ip)"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
proof -
  from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp

  from assms have "rt ≈dip invalidate rt dests"
    by (rule rt_fresh_as_inc_invalidate)
  with ⟨dip ∈ kD(rt)⟩ ⟨dip ∈ kD(invalidate rt dests)⟩ show ?thesis
    by (simp add: kD_nsqn del: invalidate_kD_inv)
      (erule(2) rt_fresh_as_nsqnr)
qed

end

```

### 3.6 Invariant proofs on individual processes

```

theory C_Seq_Invariants
imports "../.../AWN/Invariants" C_Aodv C_Aodv_Data C_Aodv_Predicates C_Fresher
begin

```

The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

```

lemma sequence_number_increases:
  "paodv i ⊨A onll ΓAODV (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
  by inv_cterms

```

```

lemma sequence_number_one_or_bigger:
  "paodv i ⊨ onl ΓAODV (λ(ξ, _). 1 ≤ sn ξ)"
  by (rule onll_step_to_invariantI [OF sequence_number_increases])
    (auto simp: σAODV_def)

```

We can get rid of the onl/onll if desired...

```

lemma sequence_number_increases':
  "paodv i ⊨A (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
  by (rule step_invariant_weakenE [OF sequence_number_increases]) (auto dest!: onllD)

```

```

lemma sequence_number_one_or_bigger':
  "paodv i ⊨ (λ(ξ, _). 1 ≤ sn ξ)"
  by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto

```

lemma sip\_in\_kD:  
 "paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l). l \in (\{PAadv-:7\} \cup \{PAadv-:5\} \cup \{PRrep-:0..PRrep-:1\} \cup \{PRreq-:0..PRreq-:3\}) \rightarrow sip \xi \in kD (rt \xi)$ )"  
 by inv\_cterms

lemma rrep\_1\_update\_changes:  
 "paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l). (l = PRrep-:1 \rightarrow rt \xi \neq update (rt \xi) (dip \xi) (dsn \xi, kno, val, hops \xi + 1, sip \xi))$ )"  
 by inv\_cterms

Proposition 7.38

lemma includes\_nhip:  
 "paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l). \forall dip \in kD(rt \xi). the (nhop (rt \xi) dip) \in kD(rt \xi)$ )"  
 proof -  
 { fix ip and  $\xi \xi' :: state$   
 assume " $\forall dip \in kD (rt \xi). the (nhop (rt \xi) dip) \in kD (rt \xi)$ "  
 and " $\xi' = \xi(|rt := update (rt \xi) ip (0, unk, val, Suc 0, ip))$ "  
 hence " $\forall dip \in kD (rt \xi).$   
   the (nhop (update (rt  $\xi$ ) ip (0, unk, val, Suc 0, ip)) dip) = ip  
    $\vee$  the (nhop (update (rt  $\xi$ ) ip (0, unk, val, Suc 0, ip)) dip)  $\in kD (rt \xi)$ "  
 by clarsimp (metis nhop\_update\_unk\_val update\_another)  
 } note one\_hop = this  
 { fix ip sip sn hops and  $\xi \xi' :: state$   
 assume " $\forall dip \in kD (rt \xi). the (nhop (rt \xi) dip) \in kD (rt \xi)$ "  
 and " $\xi' = \xi(|rt := update (rt \xi) ip (sn, kno, val, Suc hops, sip))$ "  
 and " $sip \in kD (rt \xi)$ "  
 hence "(the (nhop (update (rt  $\xi$ ) ip (sn, kno, val, Suc hops, sip)) ip) = ip  
    $\vee$  the (nhop (update (rt  $\xi$ ) ip (sn, kno, val, Suc hops, sip)) ip)  $\in kD (rt \xi)$ )  
    $\wedge$  ( $\forall dip \in kD (rt \xi).$   
     the (nhop (update (rt  $\xi$ ) ip (sn, kno, val, Suc hops, sip)) dip) = ip  
      $\vee$  the (nhop (update (rt  $\xi$ ) ip (sn, kno, val, Suc hops, sip)) dip)  $\in kD (rt \xi)$ )"  
 by (metis kD\_update\_unchanged nhop\_update\_changed update\_another)  
 } note nhip\_is\_sip = this  
 show ?thesis  
 by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf sip\_in\_kD]  
   solve: one\_hop nhip\_is\_sip)

qed

Proposition 7.4

lemma known\_destinations\_increase:  
 "paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, _), _, (\xi', _)). kD (rt \xi) \subseteq kD (rt \xi')$ )"  
 by (inv\_cterms simp add: subset\_insertI)

Proposition 7.5

lemma rreqs\_increase:  
 "paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, _), _, (\xi', _)). rreqs \xi \subseteq rreqs \xi'$ )"  
 by (inv\_cterms simp add: subset\_insertI)

lemma dests\_bigger\_than\_sqn:  
 "paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l). l \in \{PAadv-:15..PAadv-:17\} \cup \{PPkt-:7..PPkt-:9\} \cup \{PRreq-:9..PRreq-:11\} \cup \{PRreq-:17..PRreq-:19\} \cup \{PRrep-:8..PRrep-:10\} \cup \{PRerr-:1..PRerr-:4\} \cup \{PRerr-:6\} \rightarrow (\forall ip \in dom(dests \xi). ip \in kD(rt \xi) \wedge sqn (rt \xi) ip \leq the (dests \xi ip))$ )"  
 proof -

  have sqninv:  
   " $\bigwedge dests rt rsn ip.$   
   [[  $\forall ip \in dom(dests). ip \in kD(rt) \wedge sqn rt ip \leq the (dests ip); dests ip = Some rsn ]]$   
    $\implies sqn (invalidate rt dests) ip \leq rsn$ "  
   by (rule sqn\_invalidate\_in\_dests [THEN eq\_imp\_le], assumption) auto  
 have indests:

```

"∧dests rt rsn ip.
  [ [ ∃ip∈dom(dests). ip∈kD(rt) ∧ sqn rt ip ≤ the (dests ip); dests ip = Some rsn ]
    ⇒ ip∈kD(rt) ∧ sqn rt ip ≤ rsn"
  by (metis domI option.sel)
show ?thesis
  by inv_cterms
    (clarsimp split: split_if_asm option.split_asm
      elim!: sqninv indests)+
qed

```

Proposition 7.6

lemma sqns\_increase:

```

"paadv i ⊨A onll ΓAODV (λ((ξ, _), _, (ξ', _)). ∀ip. sqn (rt ξ) ip ≤ sqn (rt ξ') ip)"
proof -
  { fix ξ :: state
    assume *: "∀ip∈dom(dests ξ). ip ∈ kD (rt ξ) ∧ sqn (rt ξ) ip ≤ the (dests ξ ip)"
    have "∀ip. sqn (rt ξ) ip ≤ sqn (invalidate (rt ξ) (dests ξ)) ip"
    proof
      fix ip
      from * have "ip∉dom(dests ξ) ∨ sqn (rt ξ) ip ≤ the (dests ξ ip)" by simp
      thus "sqn (rt ξ) ip ≤ sqn (invalidate (rt ξ) (dests ξ)) ip"
        by (metis domI invalidate_sqn option.sel)
    qed
  } note solve_invalidate = this
show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf dests_bigger_than_sqn]
      simp add: solve_invalidate)
qed

```

Proposition 7.7

lemma ip\_constant:

```

"paadv i ⊨ onl ΓAODV (λ(ξ, _). ip ξ = i)"
by (inv_cterms simp add: σAODV_def)

```

Proposition 7.8

lemma sender\_ip\_valid':

```

"paadv i ⊨A onll ΓAODV (λ((ξ, _), a, _). anycast (λm. not_Pkt m → msg_sender m = ip ξ) a)"
by inv_cterms

```

lemma sender\_ip\_valid:

```

"paadv i ⊨A onll ΓAODV (λ((ξ, _), a, _). anycast (λm. not_Pkt m → msg_sender m = i) a)"
by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid'])
(auto dest!: onlD onllD)

```

lemma received\_msg\_inv:

```

"paadv i ⊨ (recvmmsg P →) onl ΓAODV (λ(ξ, l). l ∈ {PAadv-:1} → P (msg ξ))"
by inv_cterms

```

Proposition 7.9

lemma sip\_not\_ip':

```

"paadv i ⊨ (recvmmsg (λm. not_Pkt m → msg_sender m ≠ i) →) onl ΓAODV (λ(ξ, _). sip ξ ≠ ip ξ)"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]
    onl_invariant_sterms [OF aadv_wf ip_constant [THEN invariant_restrict_inD]]
    simp add: clear_locals_sip_not_ip') clarsimp+

```

lemma sip\_not\_ip:

```

"paadv i ⊨ (recvmmsg (λm. not_Pkt m → msg_sender m ≠ i) →) onl ΓAODV (λ(ξ, _). sip ξ ≠ i)"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]
    onl_invariant_sterms [OF aadv_wf ip_constant [THEN invariant_restrict_inD]]
    simp add: clear_locals_sip_not_ip') clarsimp+

```

Neither `sip_not_ip'` nor `sip_not_ip` is needed to show loop freedom.

Proposition 7.10

lemma hop\_count\_positive:  
 "paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, \_). \forall ip \in kD (rt \xi). the (dhops (rt \xi) ip) \geq 1)$ "  
 by (inv\_terms) auto

lemma rreq\_dip\_in\_vD\_dip\_eq\_ip:  
 "paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, l). (1 \in \{PRreq-:14\} \longrightarrow dip \xi \in vD(rt \xi))$   
 $\wedge (1 \in \{PRreq-:5, PRreq-:6\} \longrightarrow dip \xi = ip \xi)$   
 $\wedge (1 \in \{PRreq-:13..PRreq-:14\} \longrightarrow dip \xi \neq ip \xi))$ "  
 by inv\_terms

Proposition 7.11

lemma anycast\_msg\_zhops:  
 " $\bigwedge rreqid dip dsn dsk oip osn sip.$   
 paadv i  $\models_A$  onll  $\Gamma_{AODV} (\lambda(\_, a, \_). anycast msg_zhops a)$ "  
 proof (inv\_terms inv add:  
 onl\_invariant\_sterms [OF aadv\_wf rreq\_dip\_in\_vD\_dip\_eq\_ip [THEN invariant\_restrict\_inD]]  
 onl\_invariant\_sterms [OF aadv\_wf hop\_count\_positive [THEN invariant\_restrict\_inD]],  
 elim conjE)  
 fix l  $\xi$  a pp p' pp'  
 assume " $(\xi, pp) \in reachable (paadv i) TT$ "  
 and " $\{PRreq-:14\}unicast(\lambda\xi. the (nhop (rt \xi) (oip \xi)),$   
 $\lambda\xi. Rrep (the (dhops (rt \xi) (dip \xi))) (dip \xi) (sqn (rt \xi) (dip \xi)) (oip \xi) (ip \xi)).$   
 $p' \triangleright pp' \in sterms \Gamma_{AODV} pp$ "  
 and " $l = PRreq-:14$ "  
 and " $a = unicast (the (nhop (rt \xi) (oip \xi)))$   
 $(Rrep (the (dhops (rt \xi) (dip \xi))) (dip \xi) (sqn (rt \xi) (dip \xi)) (oip \xi) (ip \xi))$ "  
 and \*: " $\forall ip \in kD (rt \xi). Suc\ 0 \leq the (dhops (rt \xi) ip)$ "  
 and " $dip \xi \in vD (rt \xi)$ "  
 from  $\langle dip \xi \in vD (rt \xi) \rangle$  have " $dip \xi \in kD (rt \xi)$ "  
 by (rule vD\_id\_gives\_kD(1))  
 with \* have " $Suc\ 0 \leq the (dhops (rt \xi) (dip \xi))$ " ..  
 thus " $0 < the (dhops (rt \xi) (dip \xi))$ " by simp  
 qed

lemma hop\_count\_zero\_oip\_dip\_sip:  
 "paadv i  $\models$  (recvmmsg msg\_zhops  $\rightarrow$ ) onl  $\Gamma_{AODV} (\lambda(\xi, l).$   
 $(1 \in \{PAadv-:4..PAadv-:5\} \cup \{PRreq-:n/n. True\} \longrightarrow$   
 $(hops \xi = 0 \longrightarrow oip \xi = sip \xi))$   
 $\wedge$   
 $((1 \in \{PAadv-:6..PAadv-:7\} \cup \{PRrep-:n/n. True\} \longrightarrow$   
 $(hops \xi = 0 \longrightarrow dip \xi = sip \xi))))$ "  
 by (inv\_terms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]) auto

lemma osn\_rreq:  
 "paadv i  $\models$  (recvmmsg rreq\_rrep\_sn  $\rightarrow$ ) onl  $\Gamma_{AODV} (\lambda(\xi, l).$   
 $1 \in \{PAadv-:4, PAadv-:5\} \cup \{PRreq-:n/n. True\} \longrightarrow 1 \leq osn \xi)$ "  
 by (inv\_terms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]) clarsimp

lemma osn\_rreq':  
 "paadv i  $\models$  (recvmmsg  $(\lambda m. rreq_rrep\_sn\ m \wedge msg\_zhops\ m) \rightarrow$ ) onl  $\Gamma_{AODV} (\lambda(\xi, l).$   
 $1 \in \{PAadv-:4, PAadv-:5\} \cup \{PRreq-:n/n. True\} \longrightarrow 1 \leq osn \xi)$ "  
 proof (rule invariant\_weakenE [OF osn\_rreq])  
 fix a  
 assume "recvmmsg  $(\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m) a$ "  
 thus "recvmmsg rreq\_rrep\_sn a"  
 by (cases a) simp\_all  
 qed

lemma dsn\_rrep:  
 "paadv i  $\models$  (recvmmsg rreq\_rrep\_sn  $\rightarrow$ ) onl  $\Gamma_{AODV} (\lambda(\xi, l).$   
 $1 \in \{PAadv-:6, PAadv-:7\} \cup \{PRrep-:n/n. True\} \longrightarrow 1 \leq dsn \xi)$ "  
 by (inv\_terms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]) clarsimp

lemma dsn\_rrep':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ .  
 $1 \in \{PAadv-:6, PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow 1 \leq dsn\ \xi$ )"

```
proof (rule invariant_weakenE [OF dsn_rrep])
  fix a
  assume "recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ ) a"
  thus "recvmsg rreq_rrep_sn a"
    by (cases a) simp_all
qed
```

lemma hop\_count\_zero\_oip\_dip\_sip':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ .  
 $(1 \in \{PAadv-:4..PAadv-:5\} \cup \{PRreq-:n/n. True\} \rightarrow$   
 $(hops\ \xi = 0 \rightarrow oip\ \xi = sip\ \xi))$   
 $\wedge$   
 $((1 \in \{PAadv-:6..PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow$   
 $(hops\ \xi = 0 \rightarrow dip\ \xi = sip\ \xi))))$ "

```
proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])
  fix a
  assume "recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ ) a"
  thus "recvmsg msg_zhops a"
    by (cases a) simp_all
qed
```

Proposition 7.12

lemma zero\_seq\_unk\_hops\_one':

"paadv i  $\models$  (recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_)$ .  
 $\forall dip \in kD(rt\ \xi). (sqn\ (rt\ \xi)\ dip = 0 \rightarrow sqnf\ (rt\ \xi)\ dip = unk)$   
 $\wedge (sqnf\ (rt\ \xi)\ dip = unk \rightarrow the\ (dhops\ (rt\ \xi)\ dip) = 1)$   
 $\wedge (the\ (dhops\ (rt\ \xi)\ dip) = 1 \rightarrow the\ (nhop\ (rt\ \xi)\ dip) = dip))$ "

```
proof -
{ fix dip and  $\xi :: state$  and P
  assume "sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip = 0"
    and all: " $\forall ip. sqn\ (rt\ \xi)\ ip \leq sqn\ (invalidate\ (rt\ \xi)\ (dests\ \xi))\ ip$ "
    and *: "sqn (rt  $\xi$ ) dip = 0  $\implies$  P  $\xi$  dip"
  have "P  $\xi$  dip"
  proof -
    from all have "sqn (rt  $\xi$ ) dip  $\leq$  sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip" ..
    with (sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip = 0) have "sqn (rt  $\xi$ ) dip = 0" by simp
    thus "P  $\xi$  dip" by (rule *)
  qed
} note sqn_invalidate_zero [elim!] = this
```

```
{ fix dsn hops :: nat and sip oip rt and ip dip :: ip
  assume " $\forall dip \in kD(rt)$ .
    (sqn rt dip = 0  $\rightarrow$   $\pi_3$ (the (rt dip)) = unk)  $\wedge$ 
    ( $\pi_3$ (the (rt dip)) = unk  $\rightarrow$  the (dhops rt dip) = Suc 0)  $\wedge$ 
    (the (dhops rt dip) = Suc 0  $\rightarrow$  the (nhop rt dip) = dip)"
  and "hops = 0  $\rightarrow$  sip = dip"
  and "Suc 0  $\leq$  dsn"
  and "ip  $\neq$  dip  $\rightarrow$  ip  $\in$  kD(rt)"
  hence "the (dhops (update rt dip (dsn, kno, val, Suc hops, sip)) ip) = Suc 0  $\rightarrow$ 
    the (nhop (update rt dip (dsn, kno, val, Suc hops, sip)) ip) = ip"
    by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
} note prreq_ok1 [simp] = this
```

```
{ fix ip dsn hops sip oip rt dip
  assume " $\forall dip \in kD(rt)$ .
    (sqn rt dip = 0  $\rightarrow$   $\pi_3$ (the (rt dip)) = unk)  $\wedge$ 
    ( $\pi_3$ (the (rt dip)) = unk  $\rightarrow$  the (dhops rt dip) = Suc 0)  $\wedge$ 
    (the (dhops rt dip) = Suc 0  $\rightarrow$  the (nhop rt dip) = dip)"
  and "Suc 0  $\leq$  dsn"
  and "ip  $\neq$  dip  $\rightarrow$  ip  $\in$  kD(rt)"
  hence " $\pi_3$ (the (update rt dip (dsn, kno, val, Suc hops, sip) ip)) = unk  $\rightarrow$ 
    the (dhops (update rt dip (dsn, kno, val, Suc hops, sip)) ip) = Suc 0"
```



```

  by - (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)
} note prreq_ok2 [simp] = this

```

```

{ fix ip dsn hops sip oip rt dip
  assume "∀dip∈kD(rt).
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
    and "Suc 0 ≤ dsn"
    and "ip ≠ dip → ip∈kD(rt)"
  hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip)) ip = 0 →
    π3 (the (update rt dip (dsn, kno, val, Suc hops, sip) ip)) = unk"
    by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
} note prreq_ok3 [simp] = this

```

```

{ fix rt sip
  assume "∀dip∈kD rt.
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
  hence "∀dip∈kD rt.
    (sqn (update rt sip (0, unk, val, Suc 0, sip)) dip = 0 →
    π3(the (update rt sip (0, unk, val, Suc 0, sip) dip)) = unk)
    ∧ (π3(the (update rt sip (0, unk, val, Suc 0, sip) dip)) = unk →
    the (dhops (update rt sip (0, unk, val, Suc 0, sip)) dip) = Suc 0)
    ∧ (the (dhops (update rt sip (0, unk, val, Suc 0, sip)) dip) = Suc 0 →
    the (nhop (update rt sip (0, unk, val, Suc 0, sip)) dip) = dip)"
  by - (rule update_cases, simp_all add: sqnf_def sqn_def)
} note prreq_ok4 [simp] = this

```

```

have prreq_ok5 [simp]: "∧sip rt.
  π3(the (update rt sip (0, unk, val, Suc 0, sip) sip)) = unk →
  the (dhops (update rt sip (0, unk, val, Suc 0, sip)) sip) = Suc 0"
by (rule update_cases) simp_all

```

```

have prreq_ok6 [simp]: "∧sip rt.
  sqn (update rt sip (0, unk, val, Suc 0, sip)) sip = 0 →
  π3 (the (update rt sip (0, unk, val, Suc 0, sip) sip)) = unk"
by (rule update_cases) simp_all

```

show ?thesis

```

  by (inv_terms inv add: onl_invariant_sterms [OF aadv_wf hop_count_zero_oip_dip_sip']
    seq_step_invariant_sterms_TT [OF sqns_increase aadv_wf aadv_trans]
    onl_invariant_sterms [OF aadv_wf osn_rreq']
    onl_invariant_sterms [OF aadv_wf dsn_rrep']) clarsimp+

```

qed

lemma zero\_seq\_unk\_hops\_one:

```

"paadv i ⊨ (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl ΓAODV (λ(ξ, _).
  ∀dip∈kD(rt ξ). (sqn (rt ξ) dip = 0 → (sqnf (rt ξ) dip = unk
    ∧ the (dhops (rt ξ) dip) = 1
    ∧ the (nhop (rt ξ) dip) = dip)))"

```

by (rule invariant\_weakenE [OF zero\_seq\_unk\_hops\_one']) auto

lemma kD\_unk\_or\_atleast\_one:

```

"paadv i ⊨ (recvmsg rreq_rrep_sn →) onl ΓAODV (λ(ξ, l).
  ∀dip∈kD(rt ξ). π3(the (rt ξ dip)) = unk ∨ 1 ≤ π2(the (rt ξ dip)))"

```

proof -

```

{ fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2
  assume "dsk1 = unk ∨ Suc 0 ≤ dsn2"
  hence "π3(the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1) sip)) = unk
    ∨ Suc 0 ≤ sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2)) sip"
    unfolding update_def by (cases "dsk1 =unk") (clarsimp split: option.split)+
} note fromsip [simp] = this

```

```

{ fix dip sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2
  assume allkd: "∀ dip ∈ kD(rt). π3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip"
  and **: "dsk1 = unk ∨ Suc 0 ≤ dsn2"
  have "∀ dip ∈ kD(rt). π3(the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1) dip)) = unk
    ∨ Suc 0 ≤ sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2)) dip"
    (is "∀ dip ∈ kD(rt). ?prop dip")
  proof
  fix dip
  assume "dip ∈ kD(rt)"
  thus "?prop dip"
  proof (cases "dip = sip")
    assume "dip = sip"
    with ** show ?thesis
      by simp
  next
    assume "dip ≠ sip"
    with ⟨dip ∈ kD(rt)⟩ allkd show ?thesis
      by simp
  qed
  qed
} note solve_update [simp] = this

{ fix dip rt dests
  assume *: "∀ ip ∈ dom(dests). ip ∈ kD(rt) ∧ sqn rt ip ≤ the (dests ip)"
  and **: "∀ ip ∈ kD(rt). π3(the (rt ip)) = unk ∨ Suc 0 ≤ sqn rt ip"
  have "∀ dip ∈ kD(rt). π3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"
  proof
  fix dip
  assume "dip ∈ kD(rt)"
  with ** have "π3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip" ..
  thus "π3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"
  proof
    assume "π3(the (rt dip)) = unk" thus ?thesis ..
  next
    assume "Suc 0 ≤ sqn rt dip"
    have "Suc 0 ≤ sqn (invalidate rt dests) dip"
    proof (cases "dip ∈ dom(dests)")
      assume "dip ∈ dom(dests)"
      with * have "sqn rt dip ≤ the (dests dip)" by simp
      with ⟨Suc 0 ≤ sqn rt dip⟩ have "Suc 0 ≤ the (dests dip)" by simp
      with ⟨dip ∈ dom(dests)⟩ ⟨dip ∈ kD(rt)⟩ [THEN kD_Some] show ?thesis
        unfolding invalidate_def sqn_def by auto
    next
      assume "dip ∉ dom(dests)"
      with ⟨Suc 0 ≤ sqn rt dip⟩ ⟨dip ∈ kD(rt)⟩ [THEN kD_Some] show ?thesis
        unfolding invalidate_def sqn_def by auto
    qed
  thus ?thesis by (rule disjI2)
  qed
  qed
} note solve_invalidate [simp] = this

show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf dests_bigger_than_sqn
    [THEN invariant_restrict_inD]]
    onl_invariant_sterms [OF aadv_wf osn_rreq]
    onl_invariant_sterms [OF aadv_wf dsn_rrep]
    simp add: proj3_inv proj2_eq_sqn)
qed

```

Proposition 7.13

lemma rreq\_rrep\_sn\_any\_step\_invariant:

"paadv i  $\models_A$  (recvmgs rreq\_rrep\_sn  $\rightarrow$ ) onll  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_)$ . anycast rreq\_rrep\_sn a)"

proof -

```
have sqnf_kno: "paodv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ ).
                ( $1 \in \{PRreq-:14\} \longrightarrow \text{dip } \xi \in \text{kD}(\text{rt } \xi) \wedge \text{sqnf}(\text{rt } \xi) (\text{dip } \xi) = \text{kno}$ )"
  by (inv_cterms)
show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf sequence_number_one_or_bigger
                                                [THEN invariant_restrict_inD]]
      onl_invariant_sterms [OF aodv_wf kD_unk_or_atleast_one]
      onl_invariant_sterms_TT [OF aodv_wf sqnf_kno]
      onl_invariant_sterms [OF aodv_wf osn_rreq]
      onl_invariant_sterms [OF aodv_wf dsn_rrep])
    (auto simp: proj2_eq_sqn)
qed
```

Proposition 7.14

lemma rreq\_rrep\_fresh\_any\_step\_invariant:

```
"paodv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast (rreq_rrep_fresh (rt  $\xi$ )) a)"
```

proof -

```
have rreq_oip: "paodv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ ).
                ( $1 \in \{PRreq-:3, PRreq-:4, PRreq-:13, PRreq-:21\}$ 
                  $\longrightarrow \text{oip } \xi \in \text{kD}(\text{rt } \xi)$ 
                  $\wedge (\text{sqn}(\text{rt } \xi) (\text{oip } \xi) > (\text{osn } \xi)$ 
                    $\vee (\text{sqn}(\text{rt } \xi) (\text{oip } \xi) = (\text{osn } \xi)$ 
                      $\wedge \text{the}(\text{dhops}(\text{rt } \xi) (\text{oip } \xi)) \leq \text{Suc}(\text{hops } \xi)$ 
                      $\wedge \text{the}(\text{flag}(\text{rt } \xi) (\text{oip } \xi)) = \text{val}))$ )"
```

proof inv\_cterms

```
fix l  $\xi$  l' pp p'
assume "( $\xi, pp$ )  $\in$  reachable (paodv i) TT"
and "{PRreq-:2}  $\ll$   $\lambda\xi. \xi(\text{rt} :=$ 
    update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ ))  $\gg$  p'  $\in$  sterms  $\Gamma_{AODV}$  pp"
and "l' = PRreq-:3"
show "osn  $\xi$  < sqn (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )) (oip  $\xi$ )
      $\vee$  (sqn (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )) (oip  $\xi$ ) = osn  $\xi$ 
         $\wedge$  the (dhops (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )) (oip  $\xi$ ))
               $\leq$  Suc (hops  $\xi$ )
         $\wedge$  the (flag (update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )) (oip  $\xi$ ))
              = val)"
  unfolding update_def by (clarsimp split: option.split)
  (metis linorder_neqE_nat not_less)
qed
```

```
have rrep_prrep: "paodv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ ).
                 ( $1 \in \{PRrep-:2..PRrep-:5\} \longrightarrow (\text{dip } \xi \in \text{kD}(\text{rt } \xi)$ 
                    $\wedge \text{sqn}(\text{rt } \xi) (\text{dip } \xi) = \text{dsn } \xi$ 
                    $\wedge \text{the}(\text{dhops}(\text{rt } \xi) (\text{dip } \xi)) = \text{Suc}(\text{hops } \xi)$ 
                    $\wedge \text{the}(\text{flag}(\text{rt } \xi) (\text{dip } \xi)) = \text{val}$ 
                    $\wedge \text{the}(\text{nhop}(\text{rt } \xi) (\text{dip } \xi)) \in \text{kD}(\text{rt } \xi))$ )"
```

show ?thesis

```
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_oip]
    onl_invariant_sterms [OF aodv_wf rreq_dip_in_vD_dip_eq_ip]
    onl_invariant_sterms [OF aodv_wf rrep_prrep])
```

qed

Proposition 7.15

lemma rerr\_invalid\_any\_step\_invariant:

```
"paodv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast (rerr_invalid (rt  $\xi$ )) a)"
```

proof -

```
have dests_inv: "paodv i  $\models$ 
                 onl  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ ). ( $1 \in \{PAodv-:15, PPkt-:7, PRreq-:9,$ 
                   PRreq-:17, PRrep-:8, PRerr-:1}
```

```

      → (∀ ip ∈ dom(dests ξ). ip ∈ vD(rt ξ))
    ∧ (l ∈ {PAadv-:16..PAadv-:17}
        ∪ {PPkt-:8..PPkt-:9}
        ∪ {PRreq-:10..PRreq-:11}
        ∪ {PRreq-:18..PRreq-:19}
        ∪ {PRrep-:9..PRrep-:10}
        ∪ {PRerr-:2..PRerr-:4} → (∀ ip ∈ dom(dests ξ). ip ∈ iD(rt ξ)
                                  ∧ the (dests ξ ip) = sqn (rt ξ) ip))
    ∧ (l = PPkt-:12 → dip ξ ∈ iD(rt ξ)))"
  by inv_cterms (clarsimp split: split_if_asm option.split_asm simp: domIff)+
show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf dests_inv])
qed

```

Proposition 7.16

Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs  $(rip, rsn)$  is assigned to the variable  $dests$  of type  $ip \rightarrow sqn$ , or to the first argument of the function  $rerr$ , this set is a partial function, i.e., there is at most one entry  $(rip, rsn)$  for each destination  $rip$ : guaranteed by type.

lemma dests\_vD\_inc\_sqn:

```

"paadv i ⊨
  onl ΓAODV (λ(ξ, l). (l ∈ {PAadv-:15, PPkt-:7, PRreq-:9, PRreq-:17, PRrep-:8}
    → (∀ ip ∈ dom(dests ξ). ip ∈ vD(rt ξ) ∧ the (dests ξ ip) = inc (sqn (rt ξ) ip)))
    ∧ (l = PRerr-:1
      → (∀ ip ∈ dom(dests ξ). ip ∈ vD(rt ξ) ∧ the (dests ξ ip) > sqn (rt ξ) ip)))"
  by inv_cterms (clarsimp split: split_if_asm option.split_asm)+

```

Proposition 7.27

lemma route\_tables\_fresher:

```

"paadv i ⊨A (recvmsg rreq_rrep_sn →) onll ΓAODV (λ((ξ, _), _, (ξ', _)).
  ∀ dip ∈ kD(rt ξ). rt ξ ⊆dip rt ξ')"

```

proof (inv\_cterms inv add:

```

  onl_invariant_sterms [OF aadv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aadv_wf hop_count_positive [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aadv_wf osn_rreq]
  onl_invariant_sterms [OF aadv_wf dsn_rrep]
  onl_invariant_sterms [OF aadv_wf invariant_restrict_inD])

```

fix ξ pp p'

assume "(ξ, pp) ∈ reachable (paadv i) (recvmsg rreq\_rrep\_sn)"

and "{PRreq-:2}[λξ. ξ(rt := update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ))]  
p' ∈ sterms Γ<sub>AODV</sub> pp"

and "Suc 0 ≤ osn ξ"

and \*: "∀ ip ∈ kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) ip)"

show "∀ ip ∈ kD (rt ξ). rt ξ ⊆<sub>ip</sub> update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)"

proof

fix ip

assume "ip ∈ kD (rt ξ)"

moreover with \* have "1 ≤ the (dhops (rt ξ) ip)" by simp

moreover from (Suc 0 ≤ osn ξ)

have "update\_arg\_wf (osn ξ, kno, val, Suc (hops ξ), sip ξ)" ..

ultimately show "rt ξ ⊆<sub>ip</sub> update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ)"

by (rule rt\_fresher\_update)

qed

next

fix ξ pp p'

assume "(ξ, pp) ∈ reachable (paadv i) (recvmsg rreq\_rrep\_sn)"

```

    and "{PRrep-:1}[[λξ. ξ(rt := update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ))]]
      p' ∈ sterms ΓAODV pp"
    and "Suc 0 ≤ dsn ξ"
    and *: "∀ip∈kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) ip)"
show "∀ip∈kD (rt ξ). rt ξ ⊆ip update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ)"
proof
  fix ip
  assume "ip∈kD (rt ξ)"
  moreover with * have "1 ≤ the (dhops (rt ξ) ip)" by simp
  moreover from ⟨Suc 0 ≤ dsn ξ⟩
    have "update_arg_wf (dsn ξ, kno, val, Suc (hops ξ), sip ξ)" ..
  ultimately show "rt ξ ⊆ip update (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ)"
    by (rule rt_fresher_update)
qed
qed
end

```

### 3.7 The quality increases predicate

```

theory C_Quality_Increases
imports C_Aodv_Predicates C_Fresher
begin

```

```

definition quality_increases :: "state ⇒ state ⇒ bool"
where "quality_increases ξ ξ' ≡ (∀dip∈kD(rt ξ). dip ∈ kD(rt ξ') ∧ rt ξ ⊆dip rt ξ')
      ∧ (∀dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip)"

```

```

lemma quality_increasesI [intro!]:
  assumes "∧dip. dip ∈ kD(rt ξ) ⇒ dip ∈ kD(rt ξ'"
    and "∧dip. [| dip ∈ kD(rt ξ); dip ∈ kD(rt ξ') |] ⇒ rt ξ ⊆dip rt ξ'"
    and "∧dip. sqn (rt ξ) dip ≤ sqn (rt ξ') dip"
  shows "quality_increases ξ ξ'"
  unfolding quality_increases_def using assms by clarsimp

```

```

lemma quality_increasesE [elim]:
  fixes dip
  assumes "quality_increases ξ ξ'"
    and "dip∈kD(rt ξ)"
    and "[| dip ∈ kD(rt ξ'); rt ξ ⊆dip rt ξ'; sqn (rt ξ) dip ≤ sqn (rt ξ') dip |] ⇒ R dip ξ ξ'"
  shows "R dip ξ ξ'"
  using assms unfolding quality_increases_def by clarsimp

```

```

lemma quality_increases_rt_fresherD [dest]:
  fixes ip
  assumes "quality_increases ξ ξ'"
    and "ip∈kD(rt ξ)"
  shows "rt ξ ⊆ip rt ξ'"
  using assms by auto

```

```

lemma quality_increases_sqnE [elim]:
  fixes dip
  assumes "quality_increases ξ ξ'"
    and "sqn (rt ξ) dip ≤ sqn (rt ξ') dip ⇒ R dip ξ ξ'"
  shows "R dip ξ ξ'"
  using assms unfolding quality_increases_def by clarsimp

```

```

lemma quality_increases_refl [intro, simp]: "quality_increases ξ ξ"
  by rule simp_all

```

```

lemma strictly_fresher_quality_increases_right [elim]:
  fixes σ σ' dip
  assumes "rt (σ i) ⊆dip rt (σ nhip)"
    and qinc: "quality_increases (σ nhip) (σ' nhip)"

```

```

    and "dip∈kD(rt (σ nhip))"
  shows "rt (σ i) ⊆dip rt (σ' nhip)"
proof -
  from qinc have "rt (σ nhip) ⊆dip rt (σ' nhip)" using ⟨dip∈kD(rt (σ nhip))⟩
  by auto
  with ⟨rt (σ i) ⊆dip rt (σ nhip)⟩ show ?thesis ..
qed

lemma kD_quality_increases [elim]:
  assumes "i∈kD(rt ξ)"
    and "quality_increases ξ ξ'"
  shows "i∈kD(rt ξ')"
  using assms by auto

lemma kD_nsqn_quality_increases [elim]:
  assumes "i∈kD(rt ξ)"
    and "quality_increases ξ ξ'"
  shows "i∈kD(rt ξ') ∧ nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
proof -
  from assms have "i∈kD(rt ξ')" ..
  moreover with assms have "rt ξ ⊆i rt ξ'" by auto
  ultimately have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
  using ⟨i∈kD(rt ξ)⟩ by - (erule(2) rt_fresher_imp_nsqn_le)
  with ⟨i∈kD(rt ξ')⟩ show ?thesis ..
qed

lemma nsqn_quality_increases [elim]:
  assumes "i∈kD(rt ξ)"
    and "quality_increases ξ ξ'"
  shows "nsqn (rt ξ) i ≤ nsqn (rt ξ') i"
  using assms by (rule kD_nsqn_quality_increases [THEN conjunct2])

lemma kD_nsqn_quality_increases_trans [elim]:
  assumes "i∈kD(rt ξ)"
    and "s ≤ nsqn (rt ξ) i"
    and "quality_increases ξ ξ'"
  shows "i∈kD(rt ξ') ∧ s ≤ nsqn (rt ξ') i"
proof
  from ⟨i∈kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ show "i∈kD(rt ξ')" ..
next
  from ⟨i∈kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
  with ⟨s ≤ nsqn (rt ξ) i⟩ show "s ≤ nsqn (rt ξ') i" by (rule le_trans)
qed

lemma nsqn_quality_increases_nsqn_lt_lt [elim]:
  assumes "i∈kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "s < nsqn (rt ξ) i"
  shows "s < nsqn (rt ξ') i"
proof -
  from assms(1-2) have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
  with ⟨s < nsqn (rt ξ) i⟩ show "s < nsqn (rt ξ') i" by simp
qed

lemma nsqn_quality_increases_dhops [elim]:
  assumes "i∈kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "nsqn (rt ξ) i = nsqn (rt ξ') i"
  shows "the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i)"
  using assms unfolding quality_increases_def
  by (clarsimp) (drule(1) bspec, clarsimp simp: rt_fresher_def2)

lemma nsqn_quality_increases_nsqn_eq_le [elim]:
  assumes "i∈kD(rt ξ)"

```

```

    and "quality_increases  $\xi \xi'$ "
    and "s = nsqn (rt  $\xi$ ) i"
    shows "s < nsqn (rt  $\xi'$ ) i  $\vee$  (s = nsqn (rt  $\xi'$ ) i  $\wedge$  the (dhops (rt  $\xi$ ) i)  $\geq$  the (dhops (rt  $\xi'$ ) i))"
    using assms by (metis nat_less_le nsqn_quality_increases nsqn_quality_increases_dhops)

```

lemma quality\_increases\_rreq\_rrep\_props [elim]:

```

    fixes sn ip hops sip
    assumes qinc: "quality_increases ( $\sigma$  sip) ( $\sigma'$  sip)"
    and "1  $\leq$  sn"
    and *: "ip  $\in$  kD(rt ( $\sigma$  sip))  $\wedge$  sn  $\leq$  nsqn (rt ( $\sigma$  sip)) ip
             $\wedge$  (nsqn (rt ( $\sigma$  sip)) ip = sn
                 $\rightarrow$  (the (dhops (rt ( $\sigma$  sip)) ip)  $\leq$  hops
                     $\vee$  the (flag (rt ( $\sigma$  sip)) ip) = inv))"
    shows "ip  $\in$  kD(rt ( $\sigma'$  sip))  $\wedge$  sn  $\leq$  nsqn (rt ( $\sigma'$  sip)) ip
             $\wedge$  (nsqn (rt ( $\sigma'$  sip)) ip = sn
                 $\rightarrow$  (the (dhops (rt ( $\sigma'$  sip)) ip)  $\leq$  hops
                     $\vee$  the (flag (rt ( $\sigma'$  sip)) ip) = inv))"

    (is "_  $\wedge$  ?nsqnafter")

```

proof -

from \* obtain "ip  $\in$  kD(rt ( $\sigma$  sip))" and "sn  $\leq$  nsqn (rt ( $\sigma$  sip)) ip" by auto

from  $\langle$ quality\_increases ( $\sigma$  sip) ( $\sigma'$  sip) $\rangle$

have "sqn (rt ( $\sigma$  sip)) ip  $\leq$  sqn (rt ( $\sigma'$  sip)) ip" ..

from  $\langle$ quality\_increases ( $\sigma$  sip) ( $\sigma'$  sip) $\rangle$  and  $\langle$ ip  $\in$  kD (rt ( $\sigma$  sip)) $\rangle$

have "ip  $\in$  kD (rt ( $\sigma'$  sip))" ..

from  $\langle$ sn  $\leq$  nsqn (rt ( $\sigma$  sip)) ip $\rangle$  have ?nsqnafter

proof

assume "sn < nsqn (rt ( $\sigma$  sip)) ip"

also from  $\langle$ ip  $\in$  kD (rt ( $\sigma$  sip)) $\rangle$  and  $\langle$ quality\_increases ( $\sigma$  sip) ( $\sigma'$  sip) $\rangle$

have "...  $\leq$  nsqn (rt ( $\sigma'$  sip)) ip" ..

finally have "sn < nsqn (rt ( $\sigma'$  sip)) ip" .

thus ?thesis by simp

next

assume "sn = nsqn (rt ( $\sigma$  sip)) ip"

with  $\langle$ ip  $\in$  kD (rt ( $\sigma$  sip)) $\rangle$  and  $\langle$ quality\_increases ( $\sigma$  sip) ( $\sigma'$  sip) $\rangle$

have "sn < nsqn (rt ( $\sigma'$  sip)) ip

$\vee$  (sn = nsqn (rt ( $\sigma'$  sip)) ip

$\wedge$  the (dhops (rt ( $\sigma'$  sip)) ip)  $\leq$  the (dhops (rt ( $\sigma$  sip)) ip))" ..

hence "sn < nsqn (rt ( $\sigma'$  sip)) ip

$\vee$  (nsqn (rt ( $\sigma'$  sip)) ip = sn  $\wedge$  (the (dhops (rt ( $\sigma'$  sip)) ip)  $\leq$  hops  
 $\vee$  the (flag (rt ( $\sigma'$  sip)) ip) = inv))"

proof

assume "sn < nsqn (rt ( $\sigma'$  sip)) ip" thus ?thesis ..

next

assume "sn = nsqn (rt ( $\sigma'$  sip)) ip

$\wedge$  the (dhops (rt ( $\sigma$  sip)) ip)  $\geq$  the (dhops (rt ( $\sigma'$  sip)) ip)"

hence "sn = nsqn (rt ( $\sigma'$  sip)) ip"

and "the (dhops (rt ( $\sigma'$  sip)) ip)  $\leq$  the (dhops (rt ( $\sigma$  sip)) ip)" by auto

from \* and  $\langle$ sn = nsqn (rt ( $\sigma$  sip)) ip $\rangle$  have "the (dhops (rt ( $\sigma$  sip)) ip)  $\leq$  hops

$\vee$  the (flag (rt ( $\sigma$  sip)) ip) = inv"

by simp

thus ?thesis

proof

assume "the (dhops (rt ( $\sigma$  sip)) ip)  $\leq$  hops"

with  $\langle$ the (dhops (rt ( $\sigma'$  sip)) ip)  $\leq$  the (dhops (rt ( $\sigma$  sip)) ip) $\rangle$

have "the (dhops (rt ( $\sigma'$  sip)) ip)  $\leq$  hops" by simp

with  $\langle$ sn = nsqn (rt ( $\sigma'$  sip)) ip $\rangle$  show ?thesis by simp

next

assume "the (flag (rt ( $\sigma$  sip)) ip) = inv"

with  $\langle$ ip  $\in$  kD (rt ( $\sigma$  sip)) $\rangle$  have "nsqn (rt ( $\sigma$  sip)) ip = sqn (rt ( $\sigma$  sip)) ip - 1" ..

with  $\langle$ sn  $\geq$  1 $\rangle$  and  $\langle$ sn = nsqn (rt ( $\sigma$  sip)) ip $\rangle$

```

    have "sqn (rt (σ sip)) ip > 1" by simp

from ⟨ip∈kD(rt (σ' sip))⟩ show ?thesis
proof (rule vD_or_iD)
  assume "ip∈iD(rt (σ' sip))"
  hence "the (flag (rt (σ' sip)) ip) = inv" ..
  with ⟨sn = nsqn (rt (σ' sip)) ip⟩ show ?thesis
    by simp
next

  assume "ip∈vD(rt (σ' sip))"
  hence "nsqn (rt (σ' sip)) ip = sqn (rt (σ' sip)) ip" ..
  with ⟨sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip⟩
    have "nsqn (rt (σ' sip)) ip ≥ sqn (rt (σ sip)) ip" by simp

with ⟨sqn (rt (σ sip)) ip > 1⟩
  have "nsqn (rt (σ' sip)) ip > sqn (rt (σ sip)) ip - 1" by simp
with ⟨nsqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip - 1⟩
  have "nsqn (rt (σ' sip)) ip > nsqn (rt (σ sip)) ip" by simp
with ⟨sn = nsqn (rt (σ sip)) ip⟩ have "nsqn (rt (σ' sip)) ip > sn"
  by simp
  thus ?thesis ..
qed
qed
qed
thus ?thesis by (metis (mono_tags) le_cases not_le)
qed
with ⟨ip∈kD (rt (σ' sip))⟩ show "ip∈kD (rt (σ' sip)) ∧ ?nsqnafter" ..
qed

```

lemma quality\_increases\_rreq\_rrep\_props':

```

  fixes sn ip hops sip
  assumes "∀j. quality_increases (σ j) (σ' j)"
    and "1 ≤ sn"
    and *: "ip∈kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
      ∧ (nsqn (rt (σ sip)) ip = sn
        → (the (dhops (rt (σ sip)) ip) ≤ hops
          ∨ the (flag (rt (σ sip)) ip) = inv))"
  shows "ip∈kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip
    ∧ (nsqn (rt (σ' sip)) ip = sn
      → (the (dhops (rt (σ' sip)) ip) ≤ hops
        ∨ the (flag (rt (σ' sip)) ip) = inv))"

```

proof -

```

  from assms(1) have "quality_increases (σ sip) (σ' sip)" ..
  thus ?thesis using assms(2-3) by (rule quality_increases_rreq_rrep_props)
qed

```

lemma rteq\_quality\_increases:

```

  assumes "∀j. j ≠ i → quality_increases (σ j) (σ' j)"
    and "rt (σ' i) = rt (σ i)"
  shows "∀j. quality_increases (σ j) (σ' j)"
  using assms by clarsimp (metis order_refl quality_increasesI rt_fresher_refl)

```

definition msg\_fresh :: "(ip ⇒ state) ⇒ msg ⇒ bool"

where "msg\_fresh σ m ≡

```

  case m of Rreq hopsc _ _ _ oipc osnc sipc ⇒ osnc ≥ 1 ∧ (sipc ≠ oipc →
    oipc∈kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) oipc ≥ osnc
    ∧ (nsqn (rt (σ sipc)) oipc = osnc
      → (hopsc ≥ the (dhops (rt (σ sipc)) oipc)
        ∨ the (flag (rt (σ sipc)) oipc) = inv)))
  | Rrep hopsc dipc dsnc _ sipc ⇒ dsnc ≥ 1 ∧ (sipc ≠ dipc →
    dipc∈kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) dipc ≥ dsnc
    ∧ (nsqn (rt (σ sipc)) dipc = dsnc
      → (hopsc ≥ the (dhops (rt (σ sipc)) dipc)

```



```

      ∨ the (flag (rt (σ sip)) dip) = inv)))
| Rerr destsc sip ⇒ (∀ ripc ∈ dom(destsc). (ripc ∈ kD(rt (σ sip))
      ∧ the (destsc ripc) - 1 ≤ nsqn (rt (σ sip)) ripc))
| _ ⇒ True"

```

lemma msg\_fresh [simp]:

```

"∧ hops rreqid dip dsn dsk oip osn sip.
  msg_fresh σ (Rreq hops rreqid dip dsn dsk oip osn sip) =
    (osn ≥ 1 ∧ (sip ≠ oip → oip ∈ kD(rt (σ sip))
      ∧ nsqn (rt (σ sip)) oip ≥ osn
      ∧ (nsqn (rt (σ sip)) oip = osn
        → (hops ≥ the (dhops (rt (σ sip)) oip)
          ∨ the (flag (rt (σ sip)) oip) = inv))))))"
"∧ hops dip dsn oip sip. msg_fresh σ (Rrep hops dip dsn oip sip) =
  (dsn ≥ 1 ∧ (sip ≠ dip → dip ∈ kD(rt (σ sip))
    ∧ nsqn (rt (σ sip)) dip ≥ dsn
    ∧ (nsqn (rt (σ sip)) dip = dsn
      → (hops ≥ the (dhops (rt (σ sip)) dip)
        ∨ the (flag (rt (σ sip)) dip) = inv))))))"
"∧ dests sip. msg_fresh σ (Rerr dests sip) =
  (∀ ripc ∈ dom(dests). (ripc ∈ kD(rt (σ sip))
    ∧ the (dests ripc) - 1 ≤ nsqn (rt (σ sip)) ripc))"
"∧ d dip. msg_fresh σ (Newpkt d dip) = True"
"∧ d dip sip. msg_fresh σ (Pkt d dip sip) = True"
unfolding msg_fresh_def by simp_all

```

lemma msg\_fresh\_inc\_sn [simp, elim]:

```

"msg_fresh σ m ⇒ rreq_rrep_sn m"
by (cases m) simp_all

```

lemma recv\_msg\_fresh\_inc\_sn [simp, elim]:

```

"orecvmsg (msg_fresh) σ m ⇒ recvmmsg rreq_rrep_sn m"
by (cases m) simp_all

```

lemma rreq\_nsqn\_is\_fresh [simp]:

```

fixes σ msg hops rreqid dip dsn dsk oip osn sip
assumes "rreq_rrep_fresh (rt (σ sip)) (Rreq hops rreqid dip dsn dsk oip osn sip)"
and "rreq_rrep_sn (Rreq hops rreqid dip dsn dsk oip osn sip)"
shows "msg_fresh σ (Rreq hops rreqid dip dsn dsk oip osn sip)"
(is "msg_fresh σ ?msg")

```

proof -

```

let ?rt = "rt (σ sip)"
from assms(2) have "1 ≤ osn" by simp
thus ?thesis
unfolding msg_fresh_def
proof (simp only: msg.case, intro conjI impI)
  assume "sip ≠ oip"
  with assms(1) show "oip ∈ kD(?rt)" by simp
next
  assume "sip = oip"
  and "nsqn ?rt oip = osn"
  show "the (dhops ?rt oip) ≤ hops ∨ the (flag ?rt oip) = inv"
  proof (cases "oip ∈ vD(?rt)")
    assume "oip ∈ vD(?rt)"
    hence "nsqn ?rt oip = sqn ?rt oip" ..
    with (nsqn ?rt oip = osn) have "sqn ?rt oip = osn" by simp
    with assms(1) and (sip = oip) have "the (dhops ?rt oip) ≤ hops"
    by simp
    thus ?thesis ..
  next
    assume "oip ∉ vD(?rt)"
    moreover from assms(1) and (sip = oip) have "oip ∈ kD(?rt)" by simp
    ultimately have "oip ∈ iD(?rt)" by auto
    hence "the (flag ?rt oip) = inv" ..
  end

```

```

    thus ?thesis ..
  qed
next
  assume "sip ≠ oip"
  with assms(1) have "osn ≤ sqn ?rt oip" by auto
  thus "osn ≤ nsqn (rt (σ sip)) oip"
  proof (rule nat_le_eq_or_lt)
    assume "osn < sqn ?rt oip"
    hence "osn ≤ sqn ?rt oip - 1" by simp
    also have "... ≤ nsqn ?rt oip" by (rule sqn_nsqn)
    finally show "osn ≤ nsqn ?rt oip" .
  next
    assume "osn = sqn ?rt oip"
    with assms(1) and ⟨sip ≠ oip⟩ have "oip ∈ kD(?rt)"
      and "the (flag ?rt oip) = val"
      by auto
    hence "nsqn ?rt oip = sqn ?rt oip" ..
    with ⟨osn = sqn ?rt oip⟩ have "nsqn ?rt oip = osn" by simp
    thus "osn ≤ nsqn ?rt oip" by simp
  qed
qed simp
qed

lemma rrep_nsqn_is_fresh [simp]:
  fixes σ msg hops dip dsn oip sip
  assumes "rreq_rrep_fresh (rt (σ sip)) (Rrep hops dip dsn oip sip)"
    and "rreq_rrep_sn (Rrep hops dip dsn oip sip)"
  shows "msg_fresh σ (Rrep hops dip dsn oip sip)"
    (is "msg_fresh σ ?msg")
  proof -
    let ?rt = "rt (σ sip)"
    from assms have "sip ≠ dip → dip ∈ kD(?rt) ∧ sqn ?rt dip = dsn ∧ the (flag ?rt dip) = val"
      by simp
    hence "sip ≠ dip → dip ∈ kD(?rt) ∧ nsqn ?rt dip ≥ dsn"
      by clarsimp
    with assms show "msg_fresh σ ?msg"
      by clarsimp
  qed

lemma rerr_nsqn_is_fresh [simp]:
  fixes σ msg dests sip
  assumes "rerr_invalid (rt (σ sip)) (Rerr dests sip)"
  shows "msg_fresh σ (Rerr dests sip)"
    (is "msg_fresh σ ?msg")
  proof -
    let ?rt = "rt (σ sip)"
    from assms have *: "(∀ rip ∈ dom(dests). (rip ∈ iD(rt (σ sip))
      ∧ the (dests rip) = sqn (rt (σ sip)) rip))"
      by clarsimp
    have "(∀ rip ∈ dom(dests). (rip ∈ kD(rt (σ sip))
      ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip))"
      by clarsimp
  next
    proof
      fix rip
      assume "rip ∈ dom dests"
      with * have "rip ∈ iD(rt (σ sip))" and "the (dests rip) = sqn (rt (σ sip)) rip"
        by auto
      from this(2) have "the (dests rip) - 1 = sqn (rt (σ sip)) rip - 1" by simp
      also have "... ≤ nsqn (rt (σ sip)) rip" by (rule sqn_nsqn)
      finally have "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip" .
    next
      with ⟨rip ∈ iD(rt (σ sip))⟩
      show "rip ∈ kD(rt (σ sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
        by clarsimp
    qed
  qed

```

```

qed
thus "msg_fresh  $\sigma$  ?msg"
  by simp
qed

lemma quality_increases_msg_fresh [elim]:
  assumes qinc: " $\forall j$ . quality_increases ( $\sigma$  j) ( $\sigma'$  j)"
    and "msg_fresh  $\sigma$  m"
  shows "msg_fresh  $\sigma'$  m"
using assms(2)
proof (cases m)
  fix hops rreqid dip dsn dsk oip osn sip
  assume [simp]: "m = Rreq hops rreqid dip dsn dsk oip osn sip"
    and "msg_fresh  $\sigma$  m"
  then have "osn  $\geq$  1" and "sip = oip  $\vee$  (oip  $\in$  kD(rt ( $\sigma$  sip))  $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma$  sip)) oip
     $\wedge$  (nsqn (rt ( $\sigma$  sip)) oip = osn
     $\longrightarrow$  (the (dhops (rt ( $\sigma$  sip)) oip)  $\leq$  hops
     $\vee$  the (flag (rt ( $\sigma$  sip)) oip) = inv)))"

  by auto
  from this(2) show ?thesis
proof
  assume "sip = oip" with (osn  $\geq$  1) show ?thesis by simp
next
  assume "oip  $\in$  kD(rt ( $\sigma$  sip))  $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma$  sip)) oip
     $\wedge$  (nsqn (rt ( $\sigma$  sip)) oip = osn
     $\longrightarrow$  (the (dhops (rt ( $\sigma$  sip)) oip)  $\leq$  hops
     $\vee$  the (flag (rt ( $\sigma$  sip)) oip) = inv))"
  moreover from qinc have "quality_increases ( $\sigma$  sip) ( $\sigma'$  sip)" ..
  ultimately have "oip  $\in$  kD(rt ( $\sigma'$  sip))  $\wedge$  osn  $\leq$  nsqn (rt ( $\sigma'$  sip)) oip
     $\wedge$  (nsqn (rt ( $\sigma'$  sip)) oip = osn
     $\longrightarrow$  (the (dhops (rt ( $\sigma'$  sip)) oip)  $\leq$  hops
     $\vee$  the (flag (rt ( $\sigma'$  sip)) oip) = inv))"

  using (osn  $\geq$  1) by (rule quality_increases_rreq_rrep_props [rotated 2])
  with (osn  $\geq$  1) show "msg_fresh  $\sigma'$  m"
  by (clarsimp)
qed
next
  fix hops dip dsn oip sip
  assume [simp]: "m = Rrep hops dip dsn oip sip"
    and "msg_fresh  $\sigma$  m"
  then have "dsn  $\geq$  1" and "sip = dip  $\vee$  (dip  $\in$  kD(rt ( $\sigma$  sip))  $\wedge$  dsn  $\leq$  nsqn (rt ( $\sigma$  sip)) dip
     $\wedge$  (nsqn (rt ( $\sigma$  sip)) dip = dsn
     $\longrightarrow$  (the (dhops (rt ( $\sigma$  sip)) dip)  $\leq$  hops
     $\vee$  the (flag (rt ( $\sigma$  sip)) dip) = inv)))"

  by auto
  from this(2) show "?thesis"
proof
  assume "sip = dip" with (dsn  $\geq$  1) show ?thesis by simp
next
  assume "dip  $\in$  kD(rt ( $\sigma$  sip))  $\wedge$  dsn  $\leq$  nsqn (rt ( $\sigma$  sip)) dip
     $\wedge$  (nsqn (rt ( $\sigma$  sip)) dip = dsn
     $\longrightarrow$  (the (dhops (rt ( $\sigma$  sip)) dip)  $\leq$  hops
     $\vee$  the (flag (rt ( $\sigma$  sip)) dip) = inv))"
  moreover from qinc have "quality_increases ( $\sigma$  sip) ( $\sigma'$  sip)" ..
  ultimately have "dip  $\in$  kD(rt ( $\sigma'$  sip))  $\wedge$  dsn  $\leq$  nsqn (rt ( $\sigma'$  sip)) dip
     $\wedge$  (nsqn (rt ( $\sigma'$  sip)) dip = dsn
     $\longrightarrow$  (the (dhops (rt ( $\sigma'$  sip)) dip)  $\leq$  hops
     $\vee$  the (flag (rt ( $\sigma'$  sip)) dip) = inv))"

  using (dsn  $\geq$  1) by (rule quality_increases_rreq_rrep_props [rotated 2])
  with (dsn  $\geq$  1) show "msg_fresh  $\sigma'$  m"
  by clarsimp
qed
next
  fix dests sip

```

```

assume [simp]: "m = Rerr dests sip"
  and "msg_fresh  $\sigma$  m"
then have *: " $\forall rip \in \text{dom}(\text{dests}). rip \in kD(\text{rt } (\sigma \text{ sip}))$ 
               $\wedge \text{the } (\text{dests } rip) - 1 \leq \text{nsqn } (\text{rt } (\sigma \text{ sip})) \text{ rip}"$ 
  by simp
have " $\forall rip \in \text{dom}(\text{dests}). rip \in kD(\text{rt } (\sigma' \text{ sip}))$ 
       $\wedge \text{the } (\text{dests } rip) - 1 \leq \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ rip}"$ 
  proof
  fix rip
  assume "rip  $\in \text{dom}(\text{dests})"$ 
  with * have "rip  $\in kD(\text{rt } (\sigma \text{ sip}))"$  and "the (dests rip) - 1  $\leq \text{nsqn } (\text{rt } (\sigma \text{ sip})) \text{ rip}"$ 
    by - (drule(1) bspec, clarsimp)+
  moreover from qinc have "quality_increases ( $\sigma \text{ sip}$ ) ( $\sigma' \text{ sip}$ )" by simp
  ultimately show "rip  $\in kD(\text{rt } (\sigma' \text{ sip})) \wedge \text{the } (\text{dests } rip) - 1 \leq \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ rip}"$  ..
  qed
  thus ?thesis by simp
qed simp_all

```

end

### 3.8 The ‘open’ AODV model

```

theory C_OAodv
imports C_Aodv "../.../AWN/OAWN_SOS_Labels" "../.../AWN/OAWN_Convert"
begin

```

Definitions for stating and proving global network properties over individual processes.

```

definition  $\sigma_{AODV}'$  :: " $((ip \Rightarrow \text{state}) \times ((\text{state}, \text{msg}, \text{pseqp}, \text{pseqp label}) \text{seqp})) \text{set}"$ 
where " $\sigma_{AODV}' \equiv \{(\lambda i. \text{aodv\_init } i, \Gamma_{AODV} \text{ PAodv})\}"$ 

```

abbreviation opaodv

```

:: " $ip \Rightarrow ((ip \Rightarrow \text{state}) \times (\text{state}, \text{msg}, \text{pseqp}, \text{pseqp label}) \text{seqp}, \text{msg seq\_action}) \text{automaton}"$ 
where
  " $\text{opaodv } i \equiv (\mid \text{init} = \sigma_{AODV}', \text{trans} = \text{oseqp\_sos } \Gamma_{AODV} \text{ } i \mid)"$ 

```

```

lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))"
  unfolding  $\sigma_{AODV\_def}$   $\sigma_{AODV}'\_def$  by rule simp_all

```

```

lemma oaodv_control_within [simp]: "control_within  $\Gamma_{AODV}$  (init (opaodv i))"
  unfolding  $\sigma_{AODV}'\_def$  by (rule control_withinI) (auto simp del:  $\Gamma_{AODV\_simps}$ )

```

```

lemma  $\sigma_{AODV}'\_labels$  [simp]: " $(\sigma, p) \in \sigma_{AODV}' \implies \text{labels } \Gamma_{AODV} \text{ } p = \{\text{PAodv-:0}\}"$ 
  unfolding  $\sigma_{AODV}'\_def$  by simp

```

```

lemma oaodv_init_kD_empty [simp]:
  " $(\sigma, p) \in \sigma_{AODV}' \implies kD (\text{rt } (\sigma \text{ } i)) = \{\}"$ 
  unfolding  $\sigma_{AODV}'\_def$  kD_def by simp

```

```

lemma oaodv_init_vD_empty [simp]:
  " $(\sigma, p) \in \sigma_{AODV}' \implies vD (\text{rt } (\sigma \text{ } i)) = \{\}"$ 
  unfolding  $\sigma_{AODV}'\_def$  vD_def by simp

```

```

lemma oaodv_trans: "trans (opaodv i) = oseqp_sos  $\Gamma_{AODV}$  i"
  by simp

```

declare

```

oseq_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]
oseq_step_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]

```

end

### 3.9 Global invariant proofs over sequential processes

```

theory C_Global_Invariants
imports C_Seq_Invariants
  C_Aodv_Predicates
  C_Fresher
  C_Quality_Increases
  "../.../AWN/DAWN_Convert"
  C_OAodv
begin

lemma other_quality_increases [elim]:
  assumes "other_quality_increases I  $\sigma$   $\sigma'$ "
  shows " $\forall j. \text{quality\_increases } (\sigma j) (\sigma' j)$ "
  using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
  fixes m
  assumes *: "otherwith P I (orecvmsg Q)  $\sigma$   $\sigma'$  a"
  and weakenP: " $\bigwedge \sigma m. P \sigma m \implies P' \sigma m$ "
  and weakenQ: " $\bigwedge \sigma m. Q \sigma m \implies Q' \sigma m$ "
  shows "otherwith P' I (orecvmsg Q')  $\sigma$   $\sigma'$  a"
proof
  fix j
  assume "j  $\notin$  I"
  with * have "P ( $\sigma j$ ) ( $\sigma' j$ )" by auto
  thus "P' ( $\sigma j$ ) ( $\sigma' j$ )" by (rule weakenP)
next
  from * have "orecvmsg Q  $\sigma$  a" by auto
  thus "orecvmsg Q'  $\sigma$  a"
  by rule (erule weakenQ)
qed

lemma oreceived_msg_inv:
  assumes other: " $\bigwedge \sigma \sigma' m. \llbracket P \sigma m; \text{other } Q \{i\} \sigma \sigma' \rrbracket \implies P \sigma' m$ "
  and local: " $\bigwedge \sigma m. P \sigma m \implies P (\sigma(i := \sigma i(|msg := m|))) m$ "
  shows "opaodv i  $\models$  (otherwith Q {i} (orecvmsg P), other Q {i}  $\rightarrow$ )
  onl  $\Gamma_{AOADV} (\lambda(\sigma, l). l \in \{PAodv-:1\} \rightarrow P \sigma (msg (\sigma i)))$ "
proof (inv_cterms, intro impI)
  fix  $\sigma \sigma' l$ 
  assume "l = PAodv-:1  $\rightarrow$  P  $\sigma$  (msg ( $\sigma i$ ))"
  and "l = PAodv-:1"
  and "other Q {i}  $\sigma \sigma'$ "
  from this(1-2) have "P  $\sigma$  (msg ( $\sigma i$ ))" ..
  hence "P  $\sigma'$  (msg ( $\sigma i$ ))" using  $\langle$ other Q {i}  $\sigma \sigma'$  $\rangle$ 
  by (rule other)
  moreover from  $\langle$ other Q {i}  $\sigma \sigma'$  $\rangle$  have " $\sigma' i = \sigma i$ " ..
  ultimately show "P  $\sigma'$  (msg ( $\sigma' i$ ))" by simp
next
  fix  $\sigma \sigma' msg$ 
  assume "otherwith Q {i} (orecvmsg P)  $\sigma \sigma'$  (receive msg)"
  and " $\sigma' i = \sigma i(|msg := msg|)$ "
  from this(1) have "P  $\sigma$  msg"
  and " $\forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j)$ " by auto
  from this(1) have "P ( $\sigma(i := \sigma i(|msg := msg|))) msg$ " by (rule local)
  thus "P  $\sigma'$  msg"
proof (rule other)
  from  $\langle$  $\sigma' i = \sigma i(|msg := msg|)$  $\rangle$  and  $\langle$  $\forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j)$  $\rangle$ 
  show "other Q {i} ( $\sigma(i := \sigma i(|msg := msg|))) \sigma'$ "
  by - (rule otherI, auto)
qed
qed

```

(Equivalent to) Proposition 7.27

lemma local\_quality\_increases:

```
"paadv i  $\models_A$  (recvmmsg rreq_rrep_sn  $\rightarrow$ ) onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_)).$  quality_increases  $\xi \xi'$ )"
proof (rule step_invariantI)
  fix s a s'
  assume sr: "s  $\in$  reachable (paadv i) (recvmmsg rreq_rrep_sn)"
    and tr: "(s, a, s')  $\in$  trans (paadv i)"
    and rm: "recvmmsg rreq_rrep_sn a"
  from sr have srTT: "s  $\in$  reachable (paadv i) TT" ..

  from route_tables_fresher sr tr rm
  have "onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_)). \forall dip \in kD$  (rt  $\xi$ ). rt  $\xi \sqsubseteq_{dip}$  rt  $\xi'$ ) (s, a, s')"
  by (rule step_invariantD)

  moreover from known_destinations_increase srTT tr TT_True
  have "onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_)). kD$  (rt  $\xi$ )  $\subseteq$  kD (rt  $\xi'$ )) (s, a, s')"
  by (rule step_invariantD)

  moreover from sqns_increase srTT tr TT_True
  have "onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_)). \forall ip.$  sqn (rt  $\xi$ ) ip  $\leq$  sqn (rt  $\xi'$ ) ip) (s, a, s')"
  by (rule step_invariantD)

  ultimately show "onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_)).$  quality_increases  $\xi \xi'$ ) (s, a, s'"
  unfolding onll_def by auto
qed
```

lemmas olocal\_quality\_increases =

```
open_seq_step_invariant [OF local_quality_increases initiali_aadv oaadv_trans aadv_trans,
  simplified seqll_onll_swap]
```

lemma oquality\_increases:

```
"opaadv i  $\models_A$  (otherwith quality_increases {i} (orecvmmsg ( $\lambda\_.$  rreq_rrep_sn)),
  other quality_increases {i}  $\rightarrow$ )
  onll  $\Gamma_{AODV}$  ( $\lambda((\sigma, \_), \_, (\sigma', \_)). \forall j.$  quality_increases ( $\sigma j$ ) ( $\sigma' j$ ))"
```

```
(is "_  $\models_A$  (?S, _  $\rightarrow$ ) _")
```

```
proof (rule onll_ostep_invariantI, simp)
```

```
fix  $\sigma$  p l a  $\sigma'$  p' l'
```

```
assume or: "( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S (other quality_increases {i})"
```

```
and ll: "l  $\in$  labels  $\Gamma_{AODV}$  p"
```

```
and "?S  $\sigma \sigma' a$ "
```

```
and tr: "(( $\sigma, p$ ), a, ( $\sigma', p'$ ))  $\in$  oseqp_sos  $\Gamma_{AODV}$  i"
```

```
and ll': "l'  $\in$  labels  $\Gamma_{AODV}$  p'"
```

```
from this(1-3) have "orecvmmsg ( $\lambda\_.$  rreq_rrep_sn)  $\sigma a$ "
```

```
by (auto dest!: oreachable_weakenE [where QS="act (recvmmsg rreq_rrep_sn)"
  and QU="other quality_increases {i}"]
  otherwith_actionD)
```

```
with or have orw: "( $\sigma, p$ )  $\in$  oreachable (opaadv i) (act (recvmmsg rreq_rrep_sn))
  (other quality_increases {i})"
```

```
by - (erule oreachable_weakenE, auto)
```

```
with tr ll ll' and (orecvmmsg ( $\lambda\_.$  rreq_rrep_sn)  $\sigma a$ ) have "quality_increases ( $\sigma i$ ) ( $\sigma' i$ )"
```

```
by - (drule onll_ostep_invariantD [OF olocal_quality_increases], auto simp: seqll_def)
```

```
with (?S  $\sigma \sigma' a$ ) show " $\forall j.$  quality_increases ( $\sigma j$ ) ( $\sigma' j$ )"
```

```
by (auto dest!: otherwith_syncD)
```

```
qed
```

lemma rreq\_rrep\_nsqn\_fresh\_any\_step\_invariant:

```
"opaadv i  $\models_A$  (act (recvmmsg rreq_rrep_sn), other A {i}  $\rightarrow$ )
  onll  $\Gamma_{AODV}$  ( $\lambda((\sigma, \_), a, \_).$  anycast (msg_fresh  $\sigma$ ) a)"
```

```
proof (rule ostep_invariantI, simp del: act_simp)
```

```
fix  $\sigma$  p a  $\sigma'$  p'
```

```
assume or: "( $\sigma, p$ )  $\in$  oreachable (opaadv i) (act (recvmmsg rreq_rrep_sn)) (other A {i})"
```

```
and "(( $\sigma, p$ ), a, ( $\sigma', p'$ ))  $\in$  oseqp_sos  $\Gamma_{AODV}$  i"
```

```
and recv: "act (recvmmsg rreq_rrep_sn)  $\sigma \sigma' a$ "
```

```
obtain l l' where "l  $\in$  labels  $\Gamma_{AODV}$  p" and "l'  $\in$  labels  $\Gamma_{AODV}$  p'"
```

```
by (metis aadv_ex_label)
```

```

from ⟨((σ, p), a, (σ', p')) ∈ oseqp_sos ΓAODV i⟩
  have tr: "⟨((σ, p), a, (σ', p')) ∈ trans (opaadv i)⟩" by simp

have "anycast (rreq_rrep_fresh (rt (σ i))) a"
proof -
  have "opaadv i ⊨A (act (recvmsg rreq_rrep_sn), other A {i} →)
    onll ΓAODV (seqll i (λ((ξ, _), a, _). anycast (rreq_rrep_fresh (rt ξ)) a))"
  by (rule ostep_invariant_weakenE [OF
    open_seq_step_invariant [OF rreq_rrep_fresh_any_step_invariant initiali_aadv,
      simplified seqll_onll_swap]]) auto
  hence "onll ΓAODV (seqll i (λ((ξ, _), a, _). anycast (rreq_rrep_fresh (rt ξ)) a))
    ((σ, p), a, (σ', p'))"
  using or tr recv by - (erule(4) ostep_invariantE)
  thus ?thesis
  using ⟨l ∈ labels ΓAODV p⟩ and ⟨l' ∈ labels ΓAODV p'⟩ by auto
qed

moreover have "anycast (rerr_invalid (rt (σ i))) a"
proof -
  have "opaadv i ⊨A (act (recvmsg rreq_rrep_sn), other A {i} →)
    onll ΓAODV (seqll i (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a))"
  by (rule ostep_invariant_weakenE [OF
    open_seq_step_invariant [OF rerr_invalid_any_step_invariant initiali_aadv,
      simplified seqll_onll_swap]]) auto
  hence "onll ΓAODV (seqll i (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a))
    ((σ, p), a, (σ', p'))"
  using or tr recv by - (erule(4) ostep_invariantE)
  thus ?thesis
  using ⟨l ∈ labels ΓAODV p⟩ and ⟨l' ∈ labels ΓAODV p'⟩ by auto
qed

moreover have "anycast rreq_rrep_sn a"
proof -
  from or tr recv
  have "onll ΓAODV (seqll i (λ(_, a, _). anycast rreq_rrep_sn a)) ((σ, p), a, (σ', p'))"
  by (rule ostep_invariantE [OF
    open_seq_step_invariant [OF rreq_rrep_sn_any_step_invariant initiali_aadv,
      oaadv_trans aadv_trans,
      simplified seqll_onll_swap]])
  thus ?thesis
  using ⟨l ∈ labels ΓAODV p⟩ and ⟨l' ∈ labels ΓAODV p'⟩ by auto
qed

moreover have "anycast (λm. not_Pkt m → msg_sender m = i) a"
proof -
  have "opaadv i ⊨A (act (recvmsg rreq_rrep_sn), other A {i} →)
    onll ΓAODV (seqll i (λ((ξ, _), a, _). anycast (λm. not_Pkt m → msg_sender m = i) a))"
  by (rule ostep_invariant_weakenE [OF
    open_seq_step_invariant [OF sender_ip_valid initiali_aadv,
      simplified seqll_onll_swap]]) auto
  thus ?thesis using or tr recv ⟨l ∈ labels ΓAODV p⟩ and ⟨l' ∈ labels ΓAODV p'⟩
  by - (drule(3) onll_ostep_invariantD, auto)
qed

ultimately have "anycast (msg_fresh σ) a"
  by (simp_all add: anycast_def
    del: msg_fresh
    split: seq_action.split_asm msg.split_asm) simp_all
thus "onll ΓAODV (λ((σ, _), a, _). anycast (msg_fresh σ) a) ((σ, p), a, (σ', p'))"
  by auto
qed

```

lemma oreceived\_rreq\_rrep\_nsqn\_fresh\_inv:

```
"opaadv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
```

```

      other quality_increases {i} →)
      onl  $\Gamma_{AODV} (\lambda(\sigma, l). l \in \{PAodv-:1\} \longrightarrow msg\_fresh \sigma (msg (\sigma i)))$ "
proof (rule oreceived_msg_inv)
  fix  $\sigma \sigma' m$ 
  assume *: "msg_fresh  $\sigma m$ "
  and "other quality_increases {i}  $\sigma \sigma'$ "
  from this(2) have " $\forall j. quality\_increases (\sigma j) (\sigma' j)$ " ..
  thus "msg_fresh  $\sigma' m$ " using * ..
next
  fix  $\sigma m$ 
  assume "msg_fresh  $\sigma m$ "
  thus "msg_fresh ( $\sigma(i := \sigma i(|msg := m|))$ )  $m$ "
  proof (cases m)
    fix dests sip
    assume "m = Rerr dests sip"
    with (msg_fresh  $\sigma m$ ) show ?thesis by auto
  qed auto
qed

lemma oquality_increases_nsqn_fresh:
  "opaadv i  $\models_A$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onll  $\Gamma_{AODV} (\lambda((\sigma, _), _, (\sigma', _)). \forall j. quality\_increases (\sigma j) (\sigma' j))$ "
  by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
  "opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onl  $\Gamma_{AODV} (seq1 i (\lambda(\xi, l). l \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-:n \mid n. True\} \longrightarrow 1 \leq osn \xi))$ "
  by (rule oinvariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_adv]])
  (auto simp: seq1_onl_swap)

lemma rreq_sip:
  "opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onl  $\Gamma_{AODV} (\lambda(\sigma, l).
      (l \in \{PAodv-:4, PAodv-:5, PRreq-:0, PRreq-:2\} \wedge sip (\sigma i) \neq oip (\sigma i)
        \longrightarrow oip (\sigma i) \in kD(rt (\sigma (sip (\sigma i))))
          \wedge nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i)) \geq osn (\sigma i)
          \wedge (nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i)) = osn (\sigma i)
            \longrightarrow (hops (\sigma i) \geq the (dhops (rt (\sigma (sip (\sigma i)))) (oip (\sigma i)))
              \vee the (flag (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) = inv)))$ "
  (is "_  $\models$  (?S, ?U →) _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh
  adv_wf oaadv_trans]
  onl_oinvariant_sterms [OF adv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
  onl_oinvariant_sterms [OF adv_wf oosn_rreq]
  simp add: seqsimp
  simp del: One_nat_def, rule impI)
  fix  $\sigma \sigma' p l$ 
  assume " $(\sigma, p) \in oreachable (opaadv i) ?S ?U$ "
  and " $l \in labels \Gamma_{AODV} p$ "
  and pre:
    " $(l = PAodv-:4 \vee l = PAodv-:5 \vee l = PRreq-:0 \vee l = PRreq-:2) \wedge sip (\sigma i) \neq oip (\sigma i)
      \longrightarrow oip (\sigma i) \in kD (rt (\sigma (sip (\sigma i))))
        \wedge osn (\sigma i) \leq nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))
        \wedge (nsqn (rt (\sigma (sip (\sigma i)))) (oip (\sigma i)) = osn (\sigma i)
          \longrightarrow the (dhops (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) \leq hops (\sigma i)
            \vee the (flag (rt (\sigma (sip (\sigma i)))) (oip (\sigma i))) = inv)"
  and "other quality_increases {i}  $\sigma \sigma'$ "
  and hyp: " $(l=PAodv-:4 \vee l=PAodv-:5 \vee l=PRreq-:0 \vee l=PRreq-:2) \wedge sip (\sigma' i) \neq oip (\sigma' i)$ "
  (is "?labels  $\wedge sip (\sigma' i) \neq oip (\sigma' i)$ ")
  from this(4) have " $\sigma' i = \sigma i$ " ..
  with hyp have hyp': "?labels  $\wedge sip (\sigma i) \neq oip (\sigma i)$ " by simp$ 
```



```

show "oip ( $\sigma'$  i)  $\in$  kD (rt ( $\sigma'$  (sip ( $\sigma'$  i))))
   $\wedge$  osn ( $\sigma'$  i)  $\leq$  nsqn (rt ( $\sigma'$  (sip ( $\sigma'$  i)))) (oip ( $\sigma'$  i))
   $\wedge$  (nsqn (rt ( $\sigma'$  (sip ( $\sigma'$  i)))) (oip ( $\sigma'$  i)) = osn ( $\sigma'$  i)
     $\rightarrow$  the (dhops (rt ( $\sigma'$  (sip ( $\sigma'$  i)))) (oip ( $\sigma'$  i)))  $\leq$  hops ( $\sigma'$  i)
       $\vee$  the (flag (rt ( $\sigma'$  (sip ( $\sigma'$  i)))) (oip ( $\sigma'$  i))) = inv)"
proof (cases "sip ( $\sigma$  i) = i")
  assume "sip ( $\sigma$  i)  $\neq$  i"
  from <other quality_increases {i}  $\sigma$   $\sigma'$ >
    have "quality_increases ( $\sigma$  (sip ( $\sigma$  i))) ( $\sigma'$  (sip ( $\sigma$  i)))"
      by (rule otherE) (clarsimp simp: <sip ( $\sigma$  i)  $\neq$  i>)
  moreover from <( $\sigma$ , p)  $\in$  oreachable (opaadv i) ?S ?U> (l  $\in$  labels  $\Gamma_{AODV}$  p) and hyp
    have "l  $\leq$  osn ( $\sigma'$  i)"
      by (auto dest!: onl_oinvariant_weakenD [OF oosn_rreq]
          simp add: seqlsimp < $\sigma'$  i =  $\sigma$  i>)
  moreover from <sip ( $\sigma$  i)  $\neq$  i> hyp' and pre
    have "oip ( $\sigma'$  i)  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
       $\wedge$  osn ( $\sigma'$  i)  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma'$  i))
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma'$  i)) = osn ( $\sigma'$  i)
         $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma'$  i)))  $\leq$  hops ( $\sigma'$  i)
           $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma'$  i))) = inv)"
      by (auto simp: < $\sigma'$  i =  $\sigma$  i>)
    ultimately show ?thesis
      by (rule quality_increases_rreq_rrep_props)
  next
  assume "sip ( $\sigma$  i) = i" thus ?thesis
    using < $\sigma'$  i =  $\sigma$  i> hyp and pre by auto
qed
qed (auto elim!: quality_increases_rreq_rrep_props')

lemma odsn_rrep:
"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onl  $\Gamma_{AODV}$  (seq1 i ( $\lambda(\xi, l). l \in \{PAadv-:6, PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow l \leq$  dsn  $\xi$ ))"
by (rule oinvariant_weakenE [OF open_seq_invariant [OF dsn_rrep initiali_adv]])
(auto simp: seq1_onl_swap)

lemma rrep_sip:
"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onl  $\Gamma_{AODV}$  ( $\lambda(\sigma, l).$ 
    ( $l \in \{PAadv-:6, PAadv-:7, PRrep-:0, PRrep-:1\} \wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i))
     $\rightarrow$  dip ( $\sigma$  i)  $\in$  kD(rt ( $\sigma$  (sip ( $\sigma$  i))))
       $\wedge$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))  $\geq$  dsn ( $\sigma$  i)
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i)) = dsn ( $\sigma$  i)
         $\rightarrow$  (hops ( $\sigma$  i)  $\geq$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))))
           $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = inv)))"
  (is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh adv_wf
  oadv_trans]
  onl_oinvariant_sterms [OF adv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
  onl_oinvariant_sterms [OF adv_wf odsn_rrep]
  simp del: One_nat_def, rule impI)
  fix  $\sigma$   $\sigma'$  p l
  assume "( $\sigma$ , p)  $\in$  oreachable (opaadv i) ?S ?U"
  and "l  $\in$  labels  $\Gamma_{AODV}$  p"
  and pre:
    "(l = PAadv-:6  $\vee$  l = PAadv-:7  $\vee$  l = PRrep-:0  $\vee$  l = PRrep-:1)  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i)
     $\rightarrow$  dip ( $\sigma$  i)  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
       $\wedge$  dsn ( $\sigma$  i)  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i)) = dsn ( $\sigma$  i)
         $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i)))  $\leq$  hops ( $\sigma$  i)
           $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = inv)"
  and "other quality_increases {i}  $\sigma$   $\sigma'$ "
  and hyp: "(l=PAadv-:6  $\vee$  l=PAadv-:7  $\vee$  l=PRrep-:0  $\vee$  l=PRrep-:1)  $\wedge$  sip ( $\sigma'$  i)  $\neq$  dip ( $\sigma'$  i)"

```

```

      (is "?labels  $\wedge$  sip ( $\sigma'$  i)  $\neq$  dip ( $\sigma'$  i)")
from this(4) have " $\sigma' i = \sigma i$ " ..
with hyp have hyp': "?labels  $\wedge$  sip ( $\sigma i$ )  $\neq$  dip ( $\sigma i$ )" by simp
show "dip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma'$  (sip ( $\sigma' i$ ))))
   $\wedge$  dsn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ ))
   $\wedge$  (nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ ))) = dsn ( $\sigma' i$ )
   $\rightarrow$  the (dhops (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
   $\vee$  the (flag (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ ))) = inv"
proof (cases "sip ( $\sigma i$ ) = i")
  assume "sip ( $\sigma i$ )  $\neq$  i"
  from  $\langle$ other quality_increases {i}  $\sigma \sigma'$  $\rangle$ 
  have "quality_increases ( $\sigma$  (sip ( $\sigma i$ ))) ( $\sigma'$  (sip ( $\sigma' i$ )))"
  by (rule otherE) (clarsimp simp:  $\langle$ sip ( $\sigma i$ )  $\neq$  i $\rangle$ )
  moreover from  $\langle$ ( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U  $\langle$ l  $\in$  labels  $\Gamma_{AODV}$  p $\rangle$  and hyp
  have " $l \leq$  dsn ( $\sigma' i$ )"
  by (auto dest!: onl_oInvariant_weakenD [OF odsn_rrep]
      simp add: seqlsimp  $\langle$  $\sigma' i = \sigma i$  $\rangle$ )
  moreover from  $\langle$ sip ( $\sigma i$ )  $\neq$  i $\rangle$  hyp' and pre
  have "dip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma i$ ))))
     $\wedge$  dsn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma i$ )))) (dip ( $\sigma' i$ ))
     $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma i$ )))) (dip ( $\sigma' i$ ))) = dsn ( $\sigma' i$ )
     $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma i$ )))) (dip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
     $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma i$ )))) (dip ( $\sigma' i$ ))) = inv"
  by (auto simp:  $\langle$  $\sigma' i = \sigma i$  $\rangle$ )
  ultimately show ?thesis
  by (rule quality_increases_rreq_rrep_props)
next
assume "sip ( $\sigma i$ ) = i" thus ?thesis
  using  $\langle$  $\sigma' i = \sigma i$  $\rangle$  hyp and pre by auto
qed
qed (auto simp add: seqlsimp elim!: quality_increases_rreq_rrep_props')

```

lemma rerr\_sip:

```

"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onl  $\Gamma_{AODV}$  ( $\lambda(\sigma, l)$ .
    l  $\in$  {PAadv-:8, PAadv-:9, PRerr-:0, PRerr-:1}
     $\rightarrow$  ( $\forall$ rip $\in$ dom(dests ( $\sigma i$ )). rip $\in$ kD(rt ( $\sigma$  (sip ( $\sigma i$ ))))  $\wedge$ 
      the (dests ( $\sigma i$ ) rip) - 1  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma i$ )))) rip))"
(is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")

```

proof -

```

{ fix dests rip sip rsn and  $\sigma \sigma' ::$  "ip  $\Rightarrow$  state"
  assume qinc: " $\forall$ j. quality_increases ( $\sigma j$ ) ( $\sigma' j$ )"
  and *: " $\forall$ rip $\in$ dom dests. rip  $\in$  kD (rt ( $\sigma$  sip))
     $\wedge$  the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip"
  and "dests rip = Some rsn"

```

from this(3) have "rip $\in$ dom dests" by auto

with \* and  $\langle$ dests rip = Some rsn $\rangle$  have "rip $\in$ kD(rt ( $\sigma$  sip))"

and "rsn - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip"

by (auto dest!: bspec)

from qinc have "quality\_increases ( $\sigma$  sip) ( $\sigma'$  sip)" ..

have "rip  $\in$  kD(rt ( $\sigma'$  sip))  $\wedge$  rsn - 1  $\leq$  nsqn (rt ( $\sigma'$  sip)) rip"

proof

from  $\langle$ rip $\in$ kD(rt ( $\sigma$  sip)) $\rangle$  and  $\langle$ quality\_increases ( $\sigma$  sip) ( $\sigma'$  sip) $\rangle$

show "rip  $\in$  kD(rt ( $\sigma'$  sip))" ..

next

from  $\langle$ rip $\in$ kD(rt ( $\sigma$  sip)) $\rangle$  and  $\langle$ quality\_increases ( $\sigma$  sip) ( $\sigma'$  sip) $\rangle$

have "nsqn (rt ( $\sigma$  sip)) rip  $\leq$  nsqn (rt ( $\sigma'$  sip)) rip" ..

with  $\langle$ rsn - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) rip $\rangle$  show "rsn - 1  $\leq$  nsqn (rt ( $\sigma'$  sip)) rip"

by (rule le\_trans)

qed

} note partial = this

show ?thesis

```

by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
                                                    oaadv_trans]
    onl_oinvariant_sterms [OF aadv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
    other_quality_increases other_localD
    simp del: One_nat_def, intro conjI)
(clarsimp simp del: One_nat_def split: split_if_asm option.split_asm, erule(2) partial)+
qed

```

```

lemma prerr_guard: "paadv i  $\models$ 
  onl  $\Gamma_{AODV} (\lambda(\xi, l). (l = PRerr-:1
    \rightarrow (\forall ip \in \text{dom}(\text{dests } \xi). ip \in vD(\text{rt } \xi)
      \wedge \text{the } (\text{nhop } (\text{rt } \xi) ip) = \text{sip } \xi
      \wedge \text{sqn } (\text{rt } \xi) ip < \text{the } (\text{dests } \xi ip))))"$ 
  by (inv_cterms) (clarsimp split: option.split_asm split_if_asm)

```

```

lemmas odests_vD_inc_sqn =
  open_seq_invariant [OF dests_vD_inc_sqn initiali_aadv oaadv_trans aadv_trans,
    simplified seqL_onL_swap,
    THEN oinvariant_anyact]

```

```

lemmas oprerr_guard =
  open_seq_invariant [OF prerr_guard initiali_aadv oaadv_trans aadv_trans,
    simplified seqL_onL_swap,
    THEN oinvariant_anyact]

```

Proposition 7.28

```

lemma seq_compare_next_hop':
  "opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i}  $\rightarrow$ ) onl  $\Gamma_{AODV} (\lambda(\sigma, \_).
    \forall \text{dip}. \text{let } \text{nhop} = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{dip})
      \text{in } \text{dip} \in \text{kD}(\text{rt } (\sigma i)) \wedge \text{nhop} \neq \text{dip} \rightarrow
      \text{dip} \in \text{kD}(\text{rt } (\sigma \text{nhop})) \wedge \text{nsqn } (\text{rt } (\sigma i)) \text{dip} \leq \text{nsqn } (\text{rt } (\sigma \text{nhop})) \text{dip})"$ 
  (is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")
proof -

```

```

{ fix nhop and  $\sigma \sigma' :: "ip \Rightarrow \text{state}"
  assume pre: " $\forall \text{dip} \in \text{kD}(\text{rt } (\sigma i)). \text{nhop } \text{dip} \neq \text{dip} \rightarrow
    \text{dip} \in \text{kD}(\text{rt } (\sigma (\text{nhop } \text{dip}))) \wedge \text{nsqn } (\text{rt } (\sigma i)) \text{dip} \leq \text{nsqn } (\text{rt } (\sigma (\text{nhop } \text{dip}))) \text{dip}"$ 
    and qinc: " $\forall j. \text{quality\_increases } (\sigma j) (\sigma' j)"$ 
  have " $\forall \text{dip} \in \text{kD}(\text{rt } (\sigma i)). \text{nhop } \text{dip} \neq \text{dip} \rightarrow
    \text{dip} \in \text{kD}(\text{rt } (\sigma' (\text{nhop } \text{dip}))) \wedge \text{nsqn } (\text{rt } (\sigma i)) \text{dip} \leq \text{nsqn } (\text{rt } (\sigma' (\text{nhop } \text{dip}))) \text{dip}"$ 
proof (intro ballI impI)$ 
```

```

  fix dip
  assume "dip  $\in \text{kD}(\text{rt } (\sigma i))"$ 
    and "nhop dip  $\neq$  dip"
  with pre have "dip  $\in \text{kD}(\text{rt } (\sigma (\text{nhop } \text{dip})))"$ 
    and "nsqn (rt ( $\sigma i$ )) dip  $\leq$  nsqn (rt ( $\sigma (\text{nhop } \text{dip}))) \text{dip}"$ 
  by auto
  from qinc have qinc_nhop: "quality_increases ( $\sigma (\text{nhop } \text{dip})) (\sigma' (\text{nhop } \text{dip}))"$  ..
  with  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma (\text{nhop } \text{dip}))) \rangle$  have "dip  $\in \text{kD}(\text{rt } (\sigma' (\text{nhop } \text{dip})))"$  ..

```

```

moreover have "nsqn (rt ( $\sigma i$ )) dip  $\leq$  nsqn (rt ( $\sigma' (\text{nhop } \text{dip}))) \text{dip}"$ 
proof -

```

```

  from  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma (\text{nhop } \text{dip}))) \rangle$  qinc_nhop
  have "nsqn (rt ( $\sigma (\text{nhop } \text{dip}))) \text{dip} \leq \text{nsqn } (\text{rt } (\sigma' (\text{nhop } \text{dip}))) \text{dip}" ..
  with  $\langle \text{nsqn } (\text{rt } (\sigma i)) \text{dip} \leq \text{nsqn } (\text{rt } (\sigma (\text{nhop } \text{dip}))) \text{dip} \rangle$  show ?thesis
  by simp$ 
```

qed

```

ultimately show "dip  $\in \text{kD}(\text{rt } (\sigma' (\text{nhop } \text{dip})))
  \wedge \text{nsqn } (\text{rt } (\sigma i)) \text{dip} \leq \text{nsqn } (\text{rt } (\sigma' (\text{nhop } \text{dip}))) \text{dip}" ..$ 
```

qed

```

} note basic = this

```

```

{ fix nhop and  $\sigma \sigma' :: "ip \Rightarrow state"$ 
  assume pre: " $\forall dip \in kD(rt (\sigma i)). nhop dip \neq dip \longrightarrow dip \in kD(rt (\sigma (nhop dip)))$ 
     $\wedge nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (nhop dip))) dip"$ 
  and ndest: " $\forall ripc \in dom (dests (\sigma i)). ripc \in kD (rt (\sigma (sip (\sigma i))))$ 
     $\wedge the (dests (\sigma i) ripc) - 1 \leq nsqn (rt (\sigma (sip (\sigma i)))) ripc"$ 
  and issip: " $\forall ip \in dom (dests (\sigma i)). nhop ip = sip (\sigma i)"$ 
  and qinc: " $\forall j. quality\_increases (\sigma j) (\sigma' j)"$ 
  have " $\forall dip \in kD(rt (\sigma i)). nhop dip \neq dip \longrightarrow dip \in kD (rt (\sigma' (nhop dip)))$ 
     $\wedge nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"$ 
  proof (intro ballI impI)
    fix dip
    assume "dip  $\in kD(rt (\sigma i))"$ 
      and "nhop dip  $\neq dip"$ 
    with pre and qinc have "dip  $\in kD(rt (\sigma' (nhop dip)))"$ 
      and "nsqn (rt (\sigma i)) dip  $\leq nsqn (rt (\sigma' (nhop dip))) dip"$ 
    by (auto dest!: basic)

  have "nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip  $\leq nsqn (rt (\sigma' (nhop dip))) dip"$ 
  proof (cases "dip  $\in dom (dests (\sigma i))"$ )
    assume "dip  $\in dom (dests (\sigma i))"$ 
    with  $\langle dip \in kD(rt (\sigma i)) \rangle$  obtain dsn where "dests ( $\sigma i$ ) dip = Some dsn"
      by auto
    with  $\langle dip \in kD(rt (\sigma i)) \rangle$  have "nsqn (invalidate (rt ( $\sigma i$ )) (dests ( $\sigma i$ ))) dip = dsn - 1"
      by (rule nsqn_invalidate_eq)
    moreover have "dsn - 1  $\leq nsqn (rt (\sigma' (nhop dip))) dip"$ 
    proof -
      from  $\langle dests (\sigma i) dip = Some dsn \rangle$  have "the (dests ( $\sigma i$ ) dip) = dsn" by simp
      with ndest and  $\langle dip \in dom (dests (\sigma i)) \rangle$  have "dip  $\in kD (rt (\sigma (sip (\sigma i))))"$ 
        "dsn - 1  $\leq nsqn (rt (\sigma (sip (\sigma i)))) dip"$ 
      by auto
      moreover from issip and  $\langle dip \in dom (dests (\sigma i)) \rangle$  have "nhop dip = sip ( $\sigma i$ )" ..
      ultimately have "dip  $\in kD (rt (\sigma (nhop dip)))"$ 
        and "dsn - 1  $\leq nsqn (rt (\sigma (nhop dip))) dip"$  by auto
      with qinc show "dsn - 1  $\leq nsqn (rt (\sigma' (nhop dip))) dip"$ 
        by simp (metis kD_nsqn_quality_increases_trans)
    qed
    ultimately show ?thesis by simp
  next
  assume "dip  $\notin dom (dests (\sigma i))"$ 
  with  $\langle dip \in kD(rt (\sigma i)) \rangle$ 
    have "nsqn (invalidate (rt ( $\sigma i$ )) (dests ( $\sigma i$ ))) dip = nsqn (rt ( $\sigma i$ )) dip"
      by (rule nsqn_invalidate_other)
  with "nsqn (rt ( $\sigma i$ )) dip  $\leq nsqn (rt (\sigma' (nhop dip))) dip"$  show ?thesis by simp
  qed
  with  $\langle dip \in kD(rt (\sigma' (nhop dip))) \rangle$ 
    show "dip  $\in kD (rt (\sigma' (nhop dip)))$ 
       $\wedge nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"$  ..
  qed
} note basic_prerr = this

{ fix  $\sigma \sigma' :: "ip \Rightarrow state"$ 
  assume a1: " $\forall dip \in kD(rt (\sigma i)). the (nhop (rt (\sigma i)) dip) \neq dip$ 
     $\longrightarrow dip \in kD(rt (\sigma (the (nhop (rt (\sigma i)) dip))))$ 
     $\wedge nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (the (nhop (rt (\sigma i)) dip)))) dip"$ 
  and a2: " $\forall j. quality\_increases (\sigma j) (\sigma' j)"$ 
  have " $\forall dip \in kD(rt (\sigma i)).$ 
    the (nhop (update (rt ( $\sigma i$ )) (sip ( $\sigma i$ )) (0, unk, val, Suc 0, sip ( $\sigma i$ ))) dip)  $\neq dip \longrightarrow$ 
    dip  $\in kD(rt (\sigma' (the (nhop (update (rt (\sigma i)) (sip (\sigma i))$ 
      (0, unk, val, Suc 0, sip ( $\sigma i$ ))
      dip))))  $\wedge$ 
    nsqn (update (rt ( $\sigma i$ )) (sip ( $\sigma i$ )) (0, unk, val, Suc 0, sip ( $\sigma i$ ))) dip
     $\leq nsqn (rt (\sigma' (the (nhop (update (rt (\sigma i)) (sip (\sigma i))$ 
      (0, unk, val, Suc 0, sip ( $\sigma i$ ))
      dip))))
```

```

      dip" (is "∀dip∈kD(rt (σ i)). ?P dip")
proof
  fix dip
  assume "dip∈kD(rt (σ i))"
  with a1 and a2
    have "the (nhop (rt (σ i)) dip) ≠ dip → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
    by - (drule(1) basic, auto)
  thus "?P dip" by (cases "dip = sip (σ i)") auto
qed
} note nhop_update_sip = this

{ fix σ σ' oip sip osn hops
  assume pre: "∀dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip∈kD(rt (σ (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
  and qinc: "∀j. quality_increases (σ j) (σ' j)"
  and *: "sip ≠ oip → oip∈kD(rt (σ sip))
    ∧ osn ≤ nsqn (rt (σ sip)) oip
    ∧ (nsqn (rt (σ sip)) oip = osn
      → the (dhops (rt (σ sip)) oip) ≤ hops
      ∨ the (flag (rt (σ sip)) oip) = inv)"
  from pre and qinc
    have pre': "∀dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
      → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
        ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
    by (rule basic)
  have "(the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) oip) ≠ oip
    → oip∈kD(rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip)) oip))))
      ∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) oip
        ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip)) oip)))) oip)"
    (is "?nhop_not_oip → ?oip_in_kD ∧ ?nsqn_le_nsqn")
  proof (rule, split update_rt_split_asm)
    assume "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)"
    and "the (nhop (rt (σ i)) oip) ≠ oip"
    with pre' show "?oip_in_kD ∧ ?nsqn_le_nsqn" by auto
  next
    assume rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)"
    and notoip: ?nhop_not_oip
    with * qinc have ?oip_in_kD
      by (clarsimp elim!: kD_quality_increases)
    moreover with * pre qinc rtnot notoip have ?nsqn_le_nsqn
      by simp (metis kD_nsqn_quality_increases_trans)
    ultimately show "?oip_in_kD ∧ ?nsqn_le_nsqn" ..
  qed
} note update1 = this

{ fix σ σ' oip sip osn hops
  assume pre: "∀dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip∈kD(rt (σ (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
  and qinc: "∀j. quality_increases (σ j) (σ' j)"
  and *: "sip ≠ oip → oip∈kD(rt (σ sip))
    ∧ osn ≤ nsqn (rt (σ sip)) oip
    ∧ (nsqn (rt (σ sip)) oip = osn
      → the (dhops (rt (σ sip)) oip) ≤ hops
      ∨ the (flag (rt (σ sip)) oip) = inv)"
  from pre and qinc
    have pre': "∀dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
      → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
        ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
    by (rule basic)

```

```

have "∀dip∈kD(rt (σ i)).
  the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip) ≠ dip
  → dip∈kD(rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip)) dip))))
  ∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip
    ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip)) dip)))) dip"
(is "∀dip∈kD(rt (σ i)). _ → ?dip_in_kD dip ∧ ?nsqn_le_nsqn dip")
proof (intro ballI impI, split update_rt_split_asm)
  fix dip
  assume "dip∈kD(rt (σ i))"
  and "the (nhop (rt (σ i)) dip) ≠ dip"
  and "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)"
  with pre' show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip" by simp
next
  fix dip
  assume "dip∈kD(rt (σ i))"
  and notdip: "the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip)) dip) ≠ dip"
  and rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)"
  show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip"
  proof (cases "dip = oip")
    assume "dip ≠ oip"
    with pre' (dip∈kD(rt (σ i))) notdip
      show ?thesis by clarsimp
  next
    assume "dip = oip"
    with rtnot qinc (dip∈kD(rt (σ i))) notdip *
      have "?dip_in_kD dip"
        by simp (metis kD_quality_increases)
    moreover from (dip = oip) rtnot qinc (dip∈kD(rt (σ i))) notdip *
      have "?nsqn_le_nsqn dip" by simp (metis kD_nsqn_quality_increases_trans)
    ultimately show ?thesis ..
  qed
qed
} note update2 = this

have "opaadv i ⊨ (?S, ?U →) onl ΓAODV (λ(σ, _).
  ∀dip ∈ kD(rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
  → dip ∈ kD(rt (σ (the (nhop (rt (σ i)) dip))))
  ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip)"
by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
  oadv_trans]
  onl_oInvariant_sterms [OF aadv_wf odests_vD_inc_sqn]
  onl_oInvariant_sterms [OF aadv_wf oprerr_guard]
  onl_oInvariant_sterms [OF aadv_wf rreq_sip]
  onl_oInvariant_sterms [OF aadv_wf rrep_sip]
  onl_oInvariant_sterms [OF aadv_wf rerr_sip]
  other_quality_increases
  other_localD
  solve: basic basic_prerr
  simp add: seqsimp_nsqn_invalidate_nhop_update_sip
  simp del: One_nat_def)
(rule conjI, erule(2) update1, erule(2) update2)+

  thus ?thesis unfolding Let_def by auto
qed

```

Proposition 7.30

```

lemmas okD_unk_or_atleast_one =
  open_seq_invariant [OF kD_unk_or_atleast_one initiali_aadv,
    simplified seq1_onl_swap]

```

```

lemmas ozero_seq_unk_hops_one =

```

open\_seq\_invariant [OF zero\_seq\_unk\_hops\_one initiali\_aadv,  
simplified seq1\_onl\_swap]

lemma oreachable\_fresh\_okD\_unk\_or\_atleast\_one:

fixes dip  
assumes " $(\sigma, p) \in \text{oreachable } (\text{opaadv } i)$   
 $(\text{otherwith } (\text{op=}) \{i\} (\text{orecvmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m$   
 $\wedge \text{msg\_zhops } m)))$   
 $(\text{other quality\_increases } \{i\})"$   
and " $\text{dip} \in \text{kD}(\text{rt } (\sigma i))$ "  
shows " $\pi_3(\text{the } (\text{rt } (\sigma i) \text{ dip})) = \text{unk} \vee 1 \leq \pi_2(\text{the } (\text{rt } (\sigma i) \text{ dip}))$ "  
(is "?P dip")  
proof -  
have " $\exists l. l \in \text{labels } \Gamma_{AODV} p$ " by (metis aadv\_ex\_label)  
with assms(1) have " $\forall \text{dip} \in \text{kD}(\text{rt } (\sigma i)). ?P \text{ dip}$ "  
by - (drule oinvariant\_weakenD [OF okD\_unk\_or\_atleast\_one [OF oaadv\_trans aadv\_trans]],  
auto dest!: otherwith\_actionD onlD simp: seq1simp)  
with  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma i)) \rangle$  show ?thesis by simp  
qed

lemma oreachable\_fresh\_ozero\_seq\_unk\_hops\_one:

fixes dip  
assumes " $(\sigma, p) \in \text{oreachable } (\text{opaadv } i)$   
 $(\text{otherwith } (\text{op=}) \{i\} (\text{orecvmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m$   
 $\wedge \text{msg\_zhops } m)))$   
 $(\text{other quality\_increases } \{i\})"$   
and " $\text{dip} \in \text{kD}(\text{rt } (\sigma i))$ "  
shows " $\text{sqn } (\text{rt } (\sigma i)) \text{ dip} = 0 \longrightarrow$   
 $\text{sqnf } (\text{rt } (\sigma i)) \text{ dip} = \text{unk}$   
 $\wedge \text{the } (\text{dhops } (\text{rt } (\sigma i)) \text{ dip}) = 1$   
 $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip}) = \text{dip}$ "  
(is "?P dip")  
proof -  
have " $\exists l. l \in \text{labels } \Gamma_{AODV} p$ " by (metis aadv\_ex\_label)  
with assms(1) have " $\forall \text{dip} \in \text{kD}(\text{rt } (\sigma i)). ?P \text{ dip}$ "  
by - (drule oinvariant\_weakenD [OF ozero\_seq\_unk\_hops\_one [OF oaadv\_trans aadv\_trans]],  
auto dest!: onlD otherwith\_actionD simp: seq1simp)  
with  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma i)) \rangle$  show ?thesis by simp  
qed

lemma seq\_nhop\_quality\_increases':

shows " $\text{opaadv } i \models (\text{otherwith } (\text{op=}) \{i\}$   
 $(\text{orecvmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)),$   
 $\text{other quality\_increases } \{i\} \rightarrow)$   
 $\text{onl } \Gamma_{AODV} (\lambda(\sigma, \_). \forall \text{dip. let nhop} = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip})$   
 $\text{in dip} \in \text{vD } (\text{rt } (\sigma i)) \cap \text{vD } (\text{rt } (\sigma \text{ nhop}))$   
 $\wedge \text{nhop} \neq \text{dip}$   
 $\rightarrow (\text{rt } (\sigma i)) \sqsubset_{\text{dip}} (\text{rt } (\sigma \text{ nhop}))"$

(is " $\_ \models (\text{?S } i, \_ \rightarrow) \_$ ")

proof -

have weaken:

" $\bigwedge p I Q R P. p \models (\text{otherwith quality\_increases } I (\text{orecvmsg } Q), \text{other quality\_increases } I \rightarrow) P$   
 $\implies p \models (\text{otherwith } (\text{op=}) I (\text{orecvmsg } (\lambda \sigma m. Q \sigma m \wedge R \sigma m)), \text{other quality\_increases } I \rightarrow) P$ "  
by auto

{

fix  $i a$  and  $\sigma \sigma' :: "ip \Rightarrow \text{state}"$

assume a1: " $\forall \text{dip. dip} \in \text{vD}(\text{rt } (\sigma i))$

$\wedge \text{dip} \in \text{vD}(\text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip}))))$

$\wedge (\text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip})) \neq \text{dip}$

$\rightarrow \text{rt } (\sigma i) \sqsubset_{\text{dip}} \text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip})))"$

and ow: " $\text{?S } i \sigma \sigma' a$ "

have " $\forall \text{dip. dip} \in \text{vD}(\text{rt } (\sigma i))$

$\wedge \text{dip} \in \text{vD}(\text{rt } (\sigma' (\text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip}))))$

$\wedge (\text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip})) \neq \text{dip}$

```

      → rt (σ i) ⊆dip rt (σ' (the (nhop (rt (σ i)) dip)))"
proof clarify
  fix dip
  assume a2: "dip ∈ vD(rt (σ i))"
    and a3: "dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))"
    and a4: "(the (nhop (rt (σ i)) dip)) ≠ dip"
  from ow have "∀ j. j ≠ i → σ j = σ' j" by auto
  show "rt (σ i) ⊆dip rt (σ' (the (nhop (rt (σ i)) dip)))"
  proof (cases "(the (nhop (rt (σ i)) dip)) = i")
    assume "(the (nhop (rt (σ i)) dip)) = i"
    with ⟨dip ∈ vD(rt (σ i))⟩ have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))" by simp
    with a1 a2 a4 have "rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp
    with ⟨(the (nhop (rt (σ i)) dip)) = i⟩ have "rt (σ i) ⊆dip rt (σ i)" by simp
    hence False by simp
    thus ?thesis ..
  next
  assume "(the (nhop (rt (σ i)) dip)) ≠ i"
  with ⟨∀ j. j ≠ i → σ j = σ' j⟩
    have *: "σ (the (nhop (rt (σ i)) dip)) = σ' (the (nhop (rt (σ i)) dip))" by simp
  with ⟨dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))⟩
    have "dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))" by simp
  with a1 a2 a4 have "rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp
  with * show ?thesis by simp
qed
qed
} note basic = this

{ fix σ σ' a dip sip i
  assume a1: "∀ dip. dip ∈ vD(rt (σ i))
    ∧ dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))"
  and ow: "?S i σ σ' a"
  have "∀ dip. dip ∈ vD(update (rt (σ i)) sip (0, unk, val, Suc 0, sip))
    ∧ dip ∈ vD(rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip))))
    ∧ the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip) ≠ dip
    → update (rt (σ i)) sip (0, unk, val, Suc 0, sip)
      ⊆dip rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip)))"
  proof clarify
    fix dip
    assume a2: "dip ∈ vD (update (rt (σ i)) sip (0, unk, val, Suc 0, sip))"
      and a3: "dip ∈ vD(rt (σ' (the (nhop
        (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip))))"
      and a4: "the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip) ≠ dip"
    show "update (rt (σ i)) sip (0, unk, val, Suc 0, sip)
      ⊆dip rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip)))"
    proof (cases "dip = sip")
      assume "dip = sip"
      with ⟨the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip) ≠ dip⟩
        have False by simp
      thus ?thesis ..
    next
      assume [simp]: "dip ≠ sip"
      from a2 have "dip ∈ vD(rt (σ i)) ∨ dip = sip"
        by (rule vD_update_val)
      with ⟨dip ≠ sip⟩ have "dip ∈ vD(rt (σ i))" by simp
      moreover from a3 have "dip ∈ vD(rt (σ' (the (nhop (rt (σ i)) dip))))" by simp
      moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
      ultimately have "rt (σ i) ⊆dip rt (σ' (the (nhop (rt (σ i)) dip)))"
        using a1 ow by - (drule(1) basic, simp)
      with ⟨dip ≠ sip⟩ show ?thesis
        by - (erule rt_strictly_fresher_update_other, simp)
    qed
  qed
}

```



```

} note update_0_unk = this

{ fix  $\sigma$  a  $\sigma'$  nhop
  assume pre: " $\forall dip. dip \in vD(rt(\sigma i)) \wedge dip \in vD(rt(\sigma(nhop dip))) \wedge nhop dip \neq dip$ 
     $\rightarrow rt(\sigma i) \sqsubset_{dip} rt(\sigma(nhop dip))$ "
    and ow: "?S i  $\sigma$   $\sigma'$  a"
  have " $\forall dip. dip \in vD(invalidate(rt(\sigma i))(dests(\sigma i)))$ 
     $\wedge dip \in vD(rt(\sigma'(nhop dip))) \wedge nhop dip \neq dip$ 
     $\rightarrow rt(\sigma i) \sqsubset_{dip} rt(\sigma'(nhop dip))$ "
proof clarify
  fix dip
  assume "dip  $\in vD(invalidate(rt(\sigma i))(dests(\sigma i)))$ "
    and "dip  $\in vD(rt(\sigma'(nhop dip)))$ "
    and "nhop dip  $\neq dip$ "
  from this(1) have "dip  $\in vD(rt(\sigma i))$ "
    by (clarsimp dest!: vD_invalidate_vD_not_dests)
  moreover from ow have " $\forall j. j \neq i \rightarrow \sigma j = \sigma' j$ " by auto
  ultimately have "rt( $\sigma i$ )  $\sqsubset_{dip} rt(\sigma(nhop dip))$ "
    using pre  $\langle dip \in vD(rt(\sigma'(nhop dip))) \rangle$  (nhop dip  $\neq dip$ )
    by metis
  with  $\langle \forall j. j \neq i \rightarrow \sigma j = \sigma' j \rangle$  show "rt( $\sigma i$ )  $\sqsubset_{dip} rt(\sigma'(nhop dip))$ "
    by (metis rt_strictly_fresher_irefl)
qed
} note invalidate = this

{ fix  $\sigma$  a  $\sigma'$  dip oip osn sip hops i
  assume pre: " $\forall dip. dip \in vD(rt(\sigma i))$ 
     $\wedge dip \in vD(rt(\sigma(the(nhop(rt(\sigma i)) dip))))$ 
     $\wedge the(nhop(rt(\sigma i)) dip) \neq dip$ 
     $\rightarrow rt(\sigma i) \sqsubset_{dip} rt(\sigma(the(nhop(rt(\sigma i)) dip)))$ "
    and ow: "?S i  $\sigma$   $\sigma'$  a"
    and "Suc 0  $\leq$  osn"
    and a6: "sip  $\neq$  oip  $\rightarrow$  oip  $\in$  kD(rt( $\sigma sip$ ))
       $\wedge osn \leq nsqn(rt(\sigma sip)) oip$ 
       $\wedge (nsqn(rt(\sigma sip)) oip = osn$ 
         $\rightarrow the(dhops(rt(\sigma sip)) oip) \leq hops$ 
           $\vee the(flag(rt(\sigma sip)) oip) = inv$ "
    and after: " $\sigma' i = \sigma i$  (|rt := update(rt( $\sigma i$ )) oip (osn, kno, val, Suc hops, sip)|)"
  have " $\forall dip. dip \in vD(update(rt(\sigma i)) oip (osn, kno, val, Suc hops, sip))$ 
     $\wedge dip \in vD(rt(\sigma'(the(nhop(update(rt(\sigma i)) oip$ 
      (osn, kno, val, Suc hops, sip)) dip))))
     $\wedge the(nhop(update(rt(\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip) \neq dip$ 
     $\rightarrow update(rt(\sigma i)) oip (osn, kno, val, Suc hops, sip)$ 
       $\sqsubset_{dip}$ 
      rt( $\sigma'(the(nhop(update(rt(\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip)))$ "
proof clarify
  fix dip
  assume a2: "dip  $\in vD(update(rt(\sigma i)) oip (osn, kno, val, Suc(hops), sip))$ "
    and a3: "dip  $\in vD(rt(\sigma'(the(nhop(update(rt(\sigma i)) oip$ 
      (osn, kno, val, Suc hops, sip)) dip))))"
    and a4: "the(nhop(update(rt( $\sigma i$ )) oip (osn, kno, val, Suc hops, sip)) dip)  $\neq$  dip"
  from ow have a5: " $\forall j. j \neq i \rightarrow \sigma j = \sigma' j$ " by auto
  show "update(rt( $\sigma i$ )) oip (osn, kno, val, Suc hops, sip)
     $\sqsubset_{dip}$ 
    rt( $\sigma'(the(nhop(update(rt(\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip)))$ "
    (is "?rt1  $\sqsubset_{dip}$  ?rt2 dip")
proof (cases "?rt1 = rt( $\sigma i$ )")
  assume nochange [simp]:
    "update(rt( $\sigma i$ )) oip (osn, kno, val, Suc hops, sip) = rt( $\sigma i$ )"

  from after have " $\sigma' i = \sigma i$ " by simp
  with a5 have " $\forall j. \sigma j = \sigma' j$ " by metis

  from a2 have "dip  $\in vD(rt(\sigma i))$ " by simp

```

```

moreover from a3 have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))"
  using nochange and ⟨∀ j. σ j = σ' j⟩ by clarsimp
moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) ⊑dip rt (σ (the (nhop (rt (σ i)) dip)))"
  using pre by simp

hence "rt (σ i) ⊑dip rt (σ' (the (nhop (rt (σ i)) dip)))"
  using ⟨∀ j. σ j = σ' j⟩ by simp
thus "?thesis" by simp
next
assume change: "?rt1 ≠ rt (σ i)"
from after a2 have "dip ∈ kD(rt (σ' i))" by auto
show ?thesis
proof (cases "dip = oip")
  assume "dip ≠ oip"

  with a2 have "dip ∈ vD (rt (σ i))" by auto
  moreover with a3 a5 after and ⟨dip ≠ oip⟩
    have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))"
      by simp metis
  moreover from a4 and ⟨dip ≠ oip⟩ have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
  ultimately have "rt (σ i) ⊑dip rt (σ (the (nhop (rt (σ i)) dip)))"
    using pre by simp

  with after and a5 and ⟨dip ≠ oip⟩ show ?thesis
    by simp (metis rt_strictly_fresher_update_other
      rt_strictly_fresher_irefl)
next
assume "dip = oip"

with a4 and change have "sip ≠ oip" by simp
with a6 have "oip ∈ kD(rt (σ sip))"
  and "osn ≤ nsqn (rt (σ sip)) oip" by auto

from a3 change ⟨dip = oip⟩ have "oip ∈ vD(rt (σ' sip))" by simp
hence "the (flag (rt (σ' sip)) oip) = val" by simp

from ⟨oip ∈ kD(rt (σ sip))⟩
have "osn < nsqn (rt (σ' sip)) oip ∨ (osn = nsqn (rt (σ' sip)) oip
  ∧ the (dhops (rt (σ' sip)) oip) ≤ hops)"
proof
  assume "oip ∈ vD(rt (σ sip))"
  hence "the (flag (rt (σ sip)) oip) = val" by simp
  with a6 ⟨sip ≠ oip⟩ have "nsqn (rt (σ sip)) oip = osn →
    the (dhops (rt (σ sip)) oip) ≤ hops"
    by simp
  show ?thesis
proof (cases "sip = i")
  assume "sip ≠ i"
  with a5 have "σ sip = σ' sip" by simp
  with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
    and ⟨nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
  show ?thesis by auto
next
— alternative to using sip_not_ip
assume [simp]: "sip = i"
have "?rt1 = rt (σ i)"
proof (rule update_cases_kD, simp_all)
  from ⟨Suc 0 ≤ osn⟩ show "0 < osn" by simp
next
  from ⟨oip ∈ kD(rt (σ sip))⟩ and ⟨sip = i⟩ show "oip ∈ kD(rt (σ i))"
    by simp
next
  assume "sqn (rt (σ i)) oip < osn"

```

```

also from ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
  have "... ≤ nsqn (rt (σ i)) oip" by simp
also have "... ≤ sqn (rt (σ i)) oip"
  by (rule nsqn_sqn)
finally have "sqn (rt (σ i)) oip < sqn (rt (σ i)) oip" .
hence False by simp
thus "(λa. if a = oip
  then Some (osn, kno, val, Suc hops, i)
  else rt (σ i) a) = rt (σ i)" ..
next
assume "sqn (rt (σ i)) oip = osn"
  and "Suc hops < the (dhops (rt (σ i)) oip)"
from this(1) and ⟨oip ∈ vD (rt (σ sip))⟩ have "nsqn (rt (σ i)) oip = osn"
  by simp
with ⟨nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
  have "the (dhops (rt (σ i)) oip) ≤ hops" by simp
with ⟨Suc hops < the (dhops (rt (σ i)) oip)⟩ have False by simp
thus "(λa. if a = oip
  then Some (osn, kno, val, Suc hops, i)
  else rt (σ i) a) = rt (σ i)" ..
next
assume "the (flag (rt (σ i)) oip) = inv"
with ⟨the (flag (rt (σ sip)) oip) = val⟩ have False by simp
thus "(λa. if a = oip
  then Some (osn, kno, val, Suc hops, i)
  else rt (σ i) a) = rt (σ i)" ..
next
from ⟨oip ∈ kD (rt (σ sip))⟩
  show "(λa. if a = oip then Some (the (rt (σ i) oip)) else rt (σ i) a) = rt (σ i)"
  by (auto dest!: kD_Some)
qed
with change have False ..
thus ?thesis ..
qed
next
assume "oip ∈ iD (rt (σ sip))"
with ⟨the (flag (rt (σ' sip)) oip) = val⟩ and a5 have "sip = i"
  by (metis f.distinct(1) iD_flag_is_inv)
from ⟨oip ∈ iD (rt (σ sip))⟩ have "the (flag (rt (σ sip)) oip) = inv" by auto
with ⟨sip = i⟩ ⟨Suc 0 ≤ osn⟩ change after ⟨oip ∈ kD (rt (σ sip))⟩
  have "nsqn (rt (σ sip)) oip < nsqn (rt (σ' sip)) oip"
  unfolding update_def
  by (clarsimp split: option.split_asm split_if_asm)
  (auto simp: sqn_def)
with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩ have "osn < nsqn (rt (σ' sip)) oip"
  by simp
thus ?thesis ..
qed
thus ?thesis
proof
assume osnlt: "osn < nsqn (rt (σ' sip)) oip"
from ⟨dip ∈ kD (rt (σ' i))⟩ and ⟨dip = oip⟩ have "dip ∈ kD (?rt1)" by simp
moreover from a3 have "dip ∈ kD (?rt2 dip)" by simp
moreover have "nsqn ?rt1 dip < nsqn (?rt2 dip) dip"
  proof -
  have "nsqn ?rt1 oip = osn"
    by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
  also have "... < nsqn (rt (σ' sip)) oip" using osnlt .
  also have "... = nsqn (?rt2 oip) oip" by (simp add: change)
  finally show ?thesis
    using ⟨dip = oip⟩ by simp
  qed
ultimately show ?thesis
  by (rule rt_strictly_fresher_ltI)

```

```

next
  assume osneq: "osn = nsqn (rt (σ' sip)) oip ∧ the (dhops (rt (σ' sip)) oip) ≤ hops"

  have "oip ∈ kD(?rt1)" by simp
  moreover from a3 ⟨dip = oip⟩ have "oip ∈ kD(?rt2 oip)" by simp

  moreover have "nsqn ?rt1 oip = nsqn (?rt2 oip) oip"
  proof -
    from osneq have "osn = nsqn (rt (σ' sip)) oip" ..
    also have "osn = nsqn ?rt1 oip"
      by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
    also have "nsqn (rt (σ' sip)) oip = nsqn (?rt2 oip) oip"
      by (simp add: change)
    finally show ?thesis .
  qed

  moreover have "π5(the (?rt2 oip oip)) < π5(the (?rt1 oip))"
  proof -
    from osneq have "the (dhops (rt (σ' sip)) oip) ≤ hops" ..
    moreover from ⟨oip ∈ vD (rt (σ' sip))⟩ have "oip ∈ kD(rt (σ' sip))" by auto
    ultimately have "π5(the (rt (σ' sip) oip)) ≤ hops"
      by (auto simp add: proj5_eq_dhops)
    also from change after have "hops < π5(the (rt (σ' i) oip))"
      by (simp add: proj5_eq_dhops) (metis dhops_update_changed lessI)
    finally have "π5(the (rt (σ' sip) oip)) < π5(the (rt (σ' i) oip))" .
    with change after show ?thesis by simp
  qed

  ultimately have "?rt1 ⊆oip ?rt2 oip"
    by (rule rt_strictly_fresher_eqI)
  with ⟨dip = oip⟩ show ?thesis by simp
  qed

qed
qed
qed
} note rreq_rrep_update = this

have "opaadv i ⊨ (otherwith (op=) {i}) (orecvmsg (λσ m. msg_fresh σ m
                                                ∧ msg_zhops m)),
      other quality_increases {i} →)
  onl ΓAODV
  (λ(σ, _). ∀dip. dip ∈ vD (rt (σ i)) ∩ vD (rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip))))"
proof (inv_terms inv add: onl_oinvariant_sterms [OF aadv_wf rreq_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf rrep_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf rerr_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf oosn_rreq [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf odsn_rrep [THEN weaken]]
  solve: basic update_0_unk invalidate rreq_rrep_update
  simp add: seqsimp)
fix σ σ' p l
assume or: "(σ, p) ∈ oreachable (opaadv i) (?S i) (other quality_increases {i})"
  and "other quality_increases {i} σ σ'"
  and ll: "l ∈ labels ΓAODV p"
  and pre: "∀dip. dip ∈ vD (rt (σ i))
            ∧ dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))
            ∧ the (nhop (rt (σ i)) dip) ≠ dip
            → rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))"
from this(1-2)
  have or': "(σ', p) ∈ oreachable (opaadv i) (?S i) (other quality_increases {i})"
  by - (rule oreachable_other')

from or and ll have next_hop: "∀dip. let nhop = the (nhop (rt (σ i)) dip)

```

```

      in dip ∈ kD(rt (σ i)) ∧ nhip ≠ dip
      → dip ∈ kD(rt (σ nhip))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip"
by (auto dest!: onl_oinvariant_weakenD [OF seq_compare_next_hop'])

from or and ll have unk_hops_one: "∀dip∈kD (rt (σ i)). sqn (rt (σ i)) dip = 0
  → sqnf (rt (σ i)) dip = unk
  ∧ the (dhops (rt (σ i)) dip) = 1
  ∧ the (nhop (rt (σ i)) dip) = dip"
by (auto dest!: onl_oinvariant_weakenD [OF ozero_seq_unk_hops_one
  [OF oaadv_trans aadv_trans]])

  otherwith_actionD
  simp: seqsimp)

from ⟨other quality_increases {i} σ σ'⟩ have "σ' i = σ i" by auto
hence "quality_increases (σ i) (σ' i)" by auto
with ⟨other quality_increases {i} σ σ'⟩ have "∀j. quality_increases (σ j) (σ' j)"
  by - (erule otherE, metis singleton_iff)

show "∀dip. dip ∈ vD (rt (σ' i))
  ∧ dip ∈ vD (rt (σ' (the (nhop (rt (σ' i)) dip))))
  ∧ the (nhop (rt (σ' i)) dip) ≠ dip
  → rt (σ' i) ⊆dip rt (σ' (the (nhop (rt (σ' i)) dip)))"
proof clarify
  fix dip
  assume "dip∈vD(rt (σ' i))"
  and "dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))"
  and "the (nhop (rt (σ' i)) dip) ≠ dip"
  from this(1) and ⟨σ' i = σ i⟩ have "dip∈vD(rt (σ i))"
  and "dip∈kD(rt (σ i))"

  by auto

from ⟨the (nhop (rt (σ' i)) dip) ≠ dip⟩ and ⟨σ' i = σ i⟩
  have "the (nhop (rt (σ i)) dip) ≠ dip" (is "?nhip ≠ _") by simp
with ⟨dip∈kD(rt (σ i))⟩ and next_hop
  have "dip∈kD(rt (σ (?nhip)))"
  and nsqns: "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ ?nhip)) dip"
  by (auto simp: Let_def)

have "0 < sqn (rt (σ i)) dip"
  proof (rule neq0_conv [THEN iffD1, OF notI])
    assume "sqn (rt (σ i)) dip = 0"
    with ⟨dip∈kD(rt (σ i))⟩ and unk_hops_one
      have "?nhip = dip" by simp
    with ⟨?nhip ≠ dip⟩ show False ..
  qed
also have "... = nsqn (rt (σ i)) dip"
  by (rule vD_nsqn_sqn [OF ⟨dip∈vD(rt (σ i))⟩, THEN sym])
also have "... ≤ nsqn (rt (σ ?nhip)) dip"
  by (rule nsqns)
also have "... ≤ sqn (rt (σ ?nhip)) dip"
  by (rule nsqn_sqn)
finally have "0 < sqn (rt (σ ?nhip)) dip" .

have "rt (σ i) ⊆dip rt (σ' ?nhip)"
proof (cases "dip∈vD(rt (σ ?nhip))")
  assume "dip∈vD(rt (σ ?nhip))"
  with pre ⟨dip∈vD(rt (σ i))⟩ and ⟨?nhip ≠ dip⟩
    have "rt (σ i) ⊆dip rt (σ ?nhip)" by auto
  moreover from ⟨∀j. quality_increases (σ j) (σ' j)⟩
    have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
  ultimately show ?thesis
    using ⟨dip∈kD(rt (σ ?nhip))⟩
    by (rule strictly_fresher_quality_increases_right)

```

```

next
  assume "dip $\notin$ vD(rt ( $\sigma$  ?nhip))"
  with  $\langle$ dip $\in$ kD(rt ( $\sigma$  ?nhip)) $\rangle$  have "dip $\in$ iD(rt ( $\sigma$  ?nhip))" ..
  hence "the (flag (rt ( $\sigma$  ?nhip)) dip) = inv"
  by auto
  have "nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma$  ?nhip)) dip"
  by (rule nsqns)
  also from  $\langle$ dip $\in$ iD(rt ( $\sigma$  ?nhip)) $\rangle$ 
  have "... = sqn (rt ( $\sigma$  ?nhip)) dip - 1" ..
  also have "... < sqn (rt ( $\sigma'$  ?nhip)) dip"
  proof -
    from  $\langle$  $\forall j$ . quality_increases ( $\sigma$  j) ( $\sigma'$  j) $\rangle$ 
    have "quality_increases ( $\sigma$  ?nhip) ( $\sigma'$  ?nhip)" ..
    hence " $\forall ip$ . sqn (rt ( $\sigma$  ?nhip)) ip  $\leq$  sqn (rt ( $\sigma'$  ?nhip)) ip" by auto
    hence "sqn (rt ( $\sigma$  ?nhip)) dip  $\leq$  sqn (rt ( $\sigma'$  ?nhip)) dip" ..
    with  $\langle$ 0 < sqn (rt ( $\sigma$  ?nhip)) dip $\rangle$  show ?thesis by auto
  qed
  also have "... = nsqn (rt ( $\sigma'$  ?nhip)) dip"
  proof (rule vD_nsqn_sqn [THEN sym])
    from  $\langle$ dip $\in$ vD(rt ( $\sigma'$  (the (nhop (rt ( $\sigma'$  i)) dip)))) $\rangle$  and  $\langle$  $\sigma'$  i =  $\sigma$  i $\rangle$ 
    show "dip $\in$ vD(rt ( $\sigma'$  ?nhip))" by simp
  qed
  finally have "nsqn (rt ( $\sigma$  i)) dip < nsqn (rt ( $\sigma'$  ?nhip)) dip" .

  moreover from  $\langle$ dip $\in$ vD(rt ( $\sigma'$  (the (nhop (rt ( $\sigma'$  i)) dip)))) $\rangle$  and  $\langle$  $\sigma'$  i =  $\sigma$  i $\rangle$ 
  have "dip $\in$ kD(rt ( $\sigma'$  ?nhip))" by auto
  ultimately show "rt ( $\sigma$  i)  $\sqsubset_{dip}$  rt ( $\sigma'$  ?nhip)"
  using  $\langle$ dip $\in$ kD(rt ( $\sigma$  i)) $\rangle$  by - (rule rt_strictly_fresher_ltI)
  qed
  with  $\langle$  $\sigma'$  i =  $\sigma$  i $\rangle$  show "rt ( $\sigma'$  i)  $\sqsubset_{dip}$  rt ( $\sigma'$  (the (nhop (rt ( $\sigma'$  i)) dip)))"
  by simp
  qed
  qed
  thus ?thesis unfolding Let_def .
  qed

```

lemma seq\_compare\_next\_hop:

```

fixes w
shows "opaadv i  $\models$  (otherwith (op=) {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  global ( $\lambda\sigma$ .  $\forall$ dip. let nhip = the (nhop (rt ( $\sigma$  i)) dip)
    in dip  $\in$  kD(rt ( $\sigma$  i))  $\wedge$  nhip  $\neq$  dip  $\rightarrow$ 
      dip  $\in$  kD(rt ( $\sigma$  nhip))
       $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma$  nhip)) dip)"
by (rule oinvariant_weakenE [OF seq_compare_next_hop']) (auto dest!: onID)

```

lemma seq\_nhop\_quality\_increases:

```

shows "opaadv i  $\models$  (otherwith (op=) {i}
  (orecvmsg ( $\lambda\sigma$  m. msg_fresh  $\sigma$  m  $\wedge$  msg_zhops m)),
  other quality_increases {i}  $\rightarrow$ )
  global ( $\lambda\sigma$ .  $\forall$ dip. let nhip = the (nhop (rt ( $\sigma$  i)) dip)
    in dip  $\in$  vD (rt ( $\sigma$  i))  $\cap$  vD (rt ( $\sigma$  nhip))  $\wedge$  nhip  $\neq$  dip
     $\rightarrow$  (rt ( $\sigma$  i))  $\sqsubset_{dip}$  (rt ( $\sigma$  nhip)))"
by (rule oinvariant_weakenE [OF seq_nhop_quality_increases']) (auto dest!: onID)

```

end

### 3.10 Routing graphs and loop freedom

```

theory C_Loop_Freedom
imports C_Aodv_Predicates C_Fresher
begin

```

Define the central theorem that relates an invariant over network states to the absence of loops in the associate

routing graph.

**definition**

```

rt_graph :: "(ip  $\Rightarrow$  state)  $\Rightarrow$  ip  $\Rightarrow$  ip rel"
where
  "rt_graph  $\sigma$  = ( $\lambda$ dip.
    {(ip, ip') | ip ip' dsn dsk hops.
      ip  $\neq$  dip  $\wedge$  rt ( $\sigma$  ip) dip = Some (dsn, dsk, val, hops, ip')}})"

```

Given the state of a network  $\sigma$ , a routing graph for a given destination ip address  $dip$  abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

**lemma** *rt\_graphE* [elim]:

```

fixes n dip ip ip'
assumes "(ip, ip')  $\in$  rt_graph  $\sigma$  dip"
shows "ip  $\neq$  dip  $\wedge$  ( $\exists$ r. rt ( $\sigma$  ip) = r
   $\wedge$  ( $\exists$ dsn dsk hops. r dip = Some (dsn, dsk, val, hops, ip')))"
using assms unfolding rt_graph_def by auto

```

**lemma** *rt\_graph\_vD* [dest]:

```

" $\wedge$ ip ip'  $\sigma$  dip. (ip, ip')  $\in$  rt_graph  $\sigma$  dip  $\implies$  dip  $\in$  vD(rt ( $\sigma$  ip))"
unfolding rt_graph_def vD_def by auto

```

**lemma** *rt\_graph\_vD\_trans* [dest]:

```

" $\wedge$ ip ip'  $\sigma$  dip. (ip, ip')  $\in$  (rt_graph  $\sigma$  dip)+  $\implies$  dip  $\in$  vD(rt ( $\sigma$  ip))"
by (erule converse_tranclE) auto

```

**lemma** *rt\_graph\_not\_dip* [dest]:

```

" $\wedge$ ip ip'  $\sigma$  dip. (ip, ip')  $\in$  rt_graph  $\sigma$  dip  $\implies$  ip  $\neq$  dip"
unfolding rt_graph_def by auto

```

**lemma** *rt\_graph\_not\_dip\_trans* [dest]:

```

" $\wedge$ ip ip'  $\sigma$  dip. (ip, ip')  $\in$  (rt_graph  $\sigma$  dip)+  $\implies$  ip  $\neq$  dip"
by (erule converse_tranclE) auto

```

NB: the property below cannot be lifted to the transitive closure

**lemma** *rt\_graph\_nhip\_is\_nhop* [dest]:

```

" $\wedge$ ip ip'  $\sigma$  dip. (ip, ip')  $\in$  rt_graph  $\sigma$  dip  $\implies$  ip' = the (nhop (rt ( $\sigma$  ip)) dip)"
unfolding rt_graph_def by auto

```

**theorem** *inv\_to\_loop\_freedom*:

```

assumes " $\forall$ i dip. let nhip = the (nhop (rt ( $\sigma$  i)) dip)
  in dip  $\in$  vD (rt ( $\sigma$  i))  $\cap$  vD (rt ( $\sigma$  nhip))  $\wedge$  nhip  $\neq$  dip
   $\longrightarrow$  (rt ( $\sigma$  i))  $\sqsubset_{dip}$  (rt ( $\sigma$  nhip))"

```

```

shows " $\forall$ dip. irrefl ((rt_graph  $\sigma$  dip)+)"

```

```

using assms proof (intro allI)

```

```

fix  $\sigma$  :: "ip  $\Rightarrow$  state" and dip

```

```

assume inv: " $\forall$ ip dip.

```

```

  let nhip = the (nhop (rt ( $\sigma$  ip)) dip)

```

```

  in dip  $\in$  vD(rt ( $\sigma$  ip))  $\cap$  vD(rt ( $\sigma$  nhip))  $\wedge$ 

```

```

    nhip  $\neq$  dip  $\longrightarrow$  rt ( $\sigma$  ip)  $\sqsubset_{dip}$  rt ( $\sigma$  nhip)"

```

```

{ fix ip ip'

```

```

  assume "(ip, ip')  $\in$  (rt_graph  $\sigma$  dip)+"

```

```

    and "dip  $\in$  vD(rt ( $\sigma$  ip'))"

```

```

    and "ip'  $\neq$  dip"

```

```

  hence "rt ( $\sigma$  ip)  $\sqsubset_{dip}$  rt ( $\sigma$  ip')"

```

```

  proof induction

```

```

    fix nhip

```

```

    assume "(ip, nhip)  $\in$  rt_graph  $\sigma$  dip"

```

```

      and "dip  $\in$  vD(rt ( $\sigma$  nhip))"

```

```

      and "nhip  $\neq$  dip"

```

```

    from <(ip, nhip)  $\in$  rt_graph  $\sigma$  dip> have "dip  $\in$  vD(rt ( $\sigma$  ip))"

```

```

      and "nhip = the (nhop (rt ( $\sigma$  ip)) dip)"

```

```

    by auto

```

```

  from <dip  $\in$  vD(rt ( $\sigma$  ip'))> and <dip  $\in$  vD(rt ( $\sigma$  nhip))>

```

```

    have "dip ∈ vD(rt (σ ip)) ∩ vD(rt (σ nhip))" ..
  with ⟨nhip = the (nhop (rt (σ ip)) dip)⟩
    and ⟨nhip ≠ dip⟩
    and inv
    show "rt (σ ip) ⊑dip rt (σ nhip)"
    by (clarsimp simp: Let_def)
next
fix nhip nhip'
assume "(ip, nhip) ∈ (rt_graph σ dip)+"
  and "(nhip, nhip') ∈ rt_graph σ dip"
  and IH: "[| dip ∈ vD(rt (σ nhip)); nhip ≠ dip |] ⇒ rt (σ ip) ⊑dip rt (σ nhip)"
  and "dip ∈ vD(rt (σ nhip'))"
  and "nhip' ≠ dip"
from ⟨(nhip, nhip') ∈ rt_graph σ dip⟩ have 1: "dip ∈ vD(rt (σ nhip))"
  and 2: "nhip ≠ dip"
  and "nhip' = the (nhop (rt (σ nhip)) dip)"

  by auto
from 1 2 have "rt (σ ip) ⊑dip rt (σ nhip)" by (rule IH)
also have "rt (σ nhip) ⊑dip rt (σ nhip'"
proof -
  from ⟨dip ∈ vD(rt (σ nhip))⟩ and ⟨dip ∈ vD(rt (σ nhip'))⟩
  have "dip ∈ vD(rt (σ nhip)) ∩ vD(rt (σ nhip'))" ..
  with ⟨nhip' ≠ dip⟩
    and ⟨nhip' = the (nhop (rt (σ nhip)) dip)⟩
    and inv
    show "rt (σ nhip) ⊑dip rt (σ nhip'"
    by (clarsimp simp: Let_def)
qed
finally show "rt (σ ip) ⊑dip rt (σ nhip'" .
qed } note fresher = this

```

```

show "irrefl ((rt_graph σ dip)+)"
unfolding irrefl_def proof (intro allI notI)
  fix ip
  assume "(ip, ip) ∈ (rt_graph σ dip)+"
  moreover then have "dip ∈ vD(rt (σ ip))"
    and "ip ≠ dip"

  by auto
  ultimately have "rt (σ ip) ⊑dip rt (σ ip)" by (rule fresher)
  thus False by simp
qed
qed

```

end

## 3.11 Lift and transfer invariants to show loop freedom

```

theory C_Aadv_Loop_Freedom
imports "../..../AWN/OClosed_Transfer" "../..../AWN/Qmsg_Lifting" C_Global_Invariants C_Loop_Freedom
begin

```

### 3.11.1 Lift to parallel processes with queues

```

lemma par_step_no_change_on_send_or_receive:
  fixes σ s a σ' s'
  assumes "((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos ΓAODV i) (seqp_sos ΓQMSG)"
  and "a ≠ τ"
  shows "σ' i = σ i"
  using assms by (rule qmsg_no_change_on_send_or_receive)

```

```

lemma par_nhop_quality_increases:
  shows "opaadv i ⟨⟨i qmsg ⊢ (otherwith (op=) {i} (orecvmsg (λσ m.
    msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} →)

```



```

      global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
                in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
                → (rt (σ i)) ⊆dip (rt (σ nhip)))"
proof (rule lift_into_qmsg [OF seq_nhop_quality_increases])
show "opaadv i ⊢A (otherwith (op=) {i}
      (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
      other quality_increases {i} →)
      globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
proof (rule ostep_invariant_weakenE [OF oquality_increases], simp_all)
fix t :: "((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
assume "onll ΓAODV (λ((σ, _), _, (σ', _)). ∀j. quality_increases (σ j) (σ' j)) t"
thus "quality_increases (fst (fst t) i) (fst (snd (snd t)) i)"
  by (cases t) (clarsimp dest!: onllD, metis aadv_ex_label)
next
fix σ σ' a
assume "otherwith (op=) {i}
      (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)) σ σ' a"
thus "otherwith quality_increases {i} (orecvmsg (λ_. rreq_rrep_sn)) σ σ' a"
  by - (erule weaken_otherwith, auto)
qed
qed auto

lemma par_rreq_rrep_sn_quality_increases:
"opaadv i <<_i qmsg ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
proof -
have "opaadv i ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
  by (rule ostep_invariant_weakenE [OF olocal_quality_increases])
  (auto dest!: onllD seqllD elim!: aadv_ex_labelE)
hence "opaadv i <<_i qmsg ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

lemma par_rreq_rrep_nsqn_fresh_any_step:
shows "opaadv i <<_i qmsg ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ,
      other (λ_ _. True) {i} →)
      globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)"
proof -
have "opaadv i ⊢A (λσ _. (orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →)
      globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)"
proof (rule ostep_invariant_weakenE [OF rreq_rrep_nsqn_fresh_any_step_invariant])
fix t
assume "onll ΓAODV (λ((σ, _), a, _). anycast (msg_fresh σ) a) t"
thus "globala (λ(σ, a, σ'). anycast (msg_fresh σ) a) t"
  by (cases t) (clarsimp dest!: onllD, metis aadv_ex_label)
qed auto
hence "opaadv i <<_i qmsg ⊢A (λσ _. (orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →)
      globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)"
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

lemma par_anycast_msg_zhops:
shows "opaadv i <<_i qmsg ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(_, a, _). anycast msg_zhops a)"
proof -
from anycast_msg_zhops initiali_aadv oaadv_trans aadv_trans
have "opaadv i ⊢A (act TT, other (λ_ _. True) {i} →)
      seqll i (onll ΓAODV (λ(_, a, _). anycast msg_zhops a))"
  by (rule open_seq_step_invariant)
hence "opaadv i ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(_, a, _). anycast msg_zhops a)"

```

```

      globala (λ(_, a, _). anycast msg_zhops a)"
proof (rule ostep_invariant_weakenE)
  fix t :: "((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
  assume "seqll i (onll ΓAODV (λ(_, a, _). anycast msg_zhops a)) t"
  thus "globala (λ(_, a, _). anycast msg_zhops a) t"
    by (cases t) (clarsimp dest!: seqllD onllD, metis aadv_ex_label)
qed simp_all
hence "opaadv i ⟨⟨i qmsg ⊢A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i}) →⟩⟩
      globala (λ(_, a, _). anycast msg_zhops a)"
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

```

### 3.11.2 Lift to nodes

lemma node\_step\_no\_change\_on\_send\_or\_receive:

```

  assumes "((σ, NodeS i P R), a, (σ', NodeS i' P' R')) ∈ onode_sos
          (oparp_sos i (oseqp_sos ΓAODV i) (seqp_sos ΓQMSG))"
  and "a ≠ τ"
  shows "σ' i = σ i"
using assms
by (cases a) (auto elim!: par_step_no_change_on_send_or_receive)

```

lemma node\_nhop\_quality\_increases:

```

  shows "⟨ i : opaadv i ⟨⟨i qmsg : R ⟩o ⊢
        (otherwith (op=) {i})
        (oarrivmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
        other quality_increases {i}
        →⟩ global (λσ. ∀dip. let nhop = the (nhop (rt (σ i)) dip)
                          in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhop)) ∧ nhop ≠ dip
                          → (rt (σ i)) ⊑dip (rt (σ nhop)))"
  by (rule node_lift [OF par_nhop_quality_increases]) auto

```

lemma node\_quality\_increases:

```

  "⟨ i : opaadv i ⟨⟨i qmsg : R ⟩o ⊢A (λσ _. oarrivmsg (λ_. rreq_rrep_sn) σ,
                                      other (λ_ _. True) {i}) →⟩
    globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
  by (rule node_lift_step_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp

```

lemma node\_rreq\_rrep\_nsqn\_fresh\_any\_step:

```

  shows "⟨ i : opaadv i ⟨⟨i qmsg : R ⟩o ⊢A
        (λσ _. oarrivmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i}) →⟩
        globala (λ(σ, a, σ'). castmsg (msg_fresh σ) a)"
  by (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_nsqn_fresh_any_step])

```

lemma node\_anycast\_msg\_zhops:

```

  shows "⟨ i : opaadv i ⟨⟨i qmsg : R ⟩o ⊢A
        (λσ _. oarrivmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i}) →⟩
        globala (λ(_, a, _). castmsg msg_zhops a)"
  by (rule node_lift_anycast_statelessassm [OF par_anycast_msg_zhops])

```

lemma node\_silent\_change\_only:

```

  shows "⟨ i : opaadv i ⟨⟨i qmsg : Ri ⟩o ⊢A (λσ _. oarrivmsg (λ_ _. True) σ,
                                      other (λ_ _. True) {i}) →⟩
        globala (λ(σ, a, σ'). a ≠ τ → σ' i = σ i)"

```

proof (rule ostep\_invariantI, simp (no\_asm), rule impI)

```

  fix σ ζ a σ' ζ'
  assume or: "(σ, ζ) ∈ oreachable (⟨⟨i qmsg : Ri ⟩o)
            (λσ _. oarrivmsg (λ_ _. True) σ)
            (other (λ_ _. True) {i})"
  and tr: "((σ, ζ), a, (σ', ζ')) ∈ trans (⟨⟨i qmsg : Ri ⟩o)"
  and "a ≠ τn"
  from or obtain p R where "ζ = NodeS i p R"
  by - (drule node_net_state, metis)

```

```

with tr have "(( $\sigma$ , NodeS i p R), a, ( $\sigma'$ ,  $\zeta'$ ))
               $\in$  onode_sos (oparp_sos i (trans (opaadv i)) (trans qmsg))"
  by simp
thus " $\sigma' i = \sigma i$ " using (a  $\neq \tau_n$ )
  by (cases rule: onode_sos.cases)
      (auto elim: qmsg_no_change_on_send_or_receive)
qed

```

### 3.11.3 Lift to partial networks

lemma arrive\_rreq\_rrep\_nsqn\_fresh\_inc\_sn [simp]:

```

assumes "oarrivemsg ( $\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge P \sigma m$ )  $\sigma m$ "
shows "oarrivemsg ( $\lambda \_. \text{rreq\_rrep\_sn}$ )  $\sigma m$ "
using assms by (cases m) auto

```

lemma opnet\_nhop\_quality\_increases:

```

shows "opnet ( $\lambda i. \text{opaadv } i \langle \langle i \text{ qmsg} \rangle p \models$ 
      (otherwith (op=) (net_tree_ips p)
        (oarrivemsg ( $\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m$ )),
        other quality_increases (net_tree_ips p)  $\rightarrow$ )
      global ( $\lambda \sigma. \forall i \in \text{net\_tree\_ips } p. \forall \text{dip}.
        \text{let nhip} = \text{the (nhop (rt } (\sigma i)) \text{ dip})$ 
          in  $\text{dip} \in vD (\text{rt } (\sigma i)) \cap vD (\text{rt } (\sigma \text{ nhip})) \wedge \text{nhip} \neq \text{dip}$ 
           $\rightarrow (\text{rt } (\sigma i)) \sqsubset_{\text{dip}} (\text{rt } (\sigma \text{ nhip}))$ )")"

```

proof (rule pnet\_lift [OF node\_nhop\_quality\_increases])

fix i R

```

have "<i : opaadv i <<i qmsg : R>_o  $\models_A$  ( $\lambda \sigma \_ . \text{oarrivemsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma,$ 
      other ( $\lambda \_ \_ . \text{True}$ ) {i}  $\rightarrow$ ) globala ( $\lambda (\sigma, a, \sigma')$ .
      castmsg ( $\lambda m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m$ ) a)"

```

proof (rule ostep\_invariantI, simp (no\_asm))

fix  $\sigma s a \sigma' s'$

```

assume or: "( $\sigma, s$ )  $\in$  oreachable (<i : opaadv i <<i qmsg : R>_o)
            ( $\lambda \sigma \_ . \text{oarrivemsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma$ )
            (other ( $\lambda \_ \_ . \text{True}$ ) {i})"

```

and tr: "(( $\sigma, s$ ), a, ( $\sigma', s'$ ))  $\in$  trans (<i : opaadv i <<i qmsg : R>\_o)"

and am: "oarrivemsg ( $\lambda \_ . \text{rreq\_rrep\_sn}$ )  $\sigma a$ "

from or tr am have "castmsg (msg\_fresh  $\sigma$ ) a"

by (auto dest!: ostep\_invariantD [OF node\_rreq\_rrep\_nsqn\_fresh\_any\_step])

moreover from or tr am have "castmsg (msg\_zhops) a"

by (auto dest!: ostep\_invariantD [OF node\_anycast\_msg\_zhops])

ultimately show "castmsg ( $\lambda m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m$ ) a"

by (case\_tac a) auto

qed

```

thus "<i : opaadv i <<i qmsg : R>_o  $\models_A$ 
      ( $\lambda \sigma \_ . \text{oarrivemsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) \sigma,$ 
      other quality_increases {i}  $\rightarrow$ ) globala ( $\lambda (\sigma, a, \_)$ .
      castmsg ( $\lambda m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m$ ) a)"

```

by rule auto

next

fix i R

```

show "<i : opaadv i <<i qmsg : R>_o  $\models_A$ 
      ( $\lambda \sigma \_ . \text{oarrivemsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) \sigma,$ 
      other quality_increases {i}  $\rightarrow$ ) globala ( $\lambda (\sigma, a, \sigma')$ .
      a  $\neq \tau \wedge (\forall i d. a \neq i:\text{deliver}(d)) \rightarrow \sigma i = \sigma' i$ )"

```

by (rule ostep\_invariant\_weakenE [OF node\_silent\_change\_only]) auto

next

fix i R

```

show "<i : opaadv i <<i qmsg : R>_o  $\models_A$ 
      ( $\lambda \sigma \_ . \text{oarrivemsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) \sigma,$ 
      other quality_increases {i}  $\rightarrow$ ) globala ( $\lambda (\sigma, a, \sigma')$ .
      a =  $\tau \vee (\exists d. a = i:\text{deliver}(d)) \rightarrow \text{quality\_increases } (\sigma i) (\sigma' i)$ )"

```

by (rule ostep\_invariant\_weakenE [OF node\_quality\_increases]) auto

qed simp\_all

### 3.11.4 Lift to closed networks

lemma onet\_nhop\_quality\_increases:

```
shows "oclosed (opnet (λi. opaadv i ⟨⟨i qmsg⟩ p⟩)
  ⊨ (λ_ _ . True, other quality_increases (net_tree_ips p) →)
    global (λσ. ∀i∈net_tree_ips p. ∀dip.
      let nhop = the (nhop (rt (σ i)) dip)
      in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhop)) ∧ nhop ≠ dip
      → (rt (σ i)) ⊑dip (rt (σ nhop)))")"
```

(is "\_ ⊨ (\_, ?U →) ?inv")

proof (rule inclosed\_closed)

from opnet\_nhop\_quality\_increases

```
show "opnet (λi. opaadv i ⟨⟨i qmsg⟩ p⟩)
  ⊨ (otherwith (op=) (net_tree_ips p) inclosed, ?U →) ?inv"
```

proof (rule oinvariant\_weakenE)

```
fix σ σ' :: "ip ⇒ state" and a :: "msg node_action"
assume "otherwith (op=) (net_tree_ips p) inclosed σ σ' a"
thus "otherwith (op=) (net_tree_ips p)
  (oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m)) σ σ' a"
```

proof (rule otherwithEI)

```
fix σ :: "ip ⇒ state" and a :: "msg node_action"
assume "inclosed σ a"
thus "oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ a"
```

proof (cases a)

```
fix ii ni ms
assume "a = ii-ni:arrive(ms)"
moreover with (inclosed σ a) obtain d di where "ms = newpkt(d, di)"
by (cases ms) auto
ultimately show ?thesis by simp
```

qed simp\_all

qed

qed

qed

### 3.11.5 Transfer into the standard model

interpretation aadv\_openproc: openproc paadv opaadv id

where "aadv\_openproc.initmissing = initmissing"

proof -

show "openproc paadv opaadv id"

proof unfold\_locales

fix i :: ip

have "{(σ, ζ). (σ i, ζ) ∈ σ<sub>AADV</sub> i ∧ (∀j. j ≠ i → σ j ∈ fst ' σ<sub>AADV</sub> j)} ⊆ σ<sub>AADV</sub>'"

unfolding σ<sub>AADV</sub>\_def σ<sub>AADV</sub>'\_def

proof (rule equalityD1)

```
show "∧f p. {(σ, ζ). (σ i, ζ) ∈ {(f i, p)} ∧ (∀j. j ≠ i
  → σ j ∈ fst ' {(f j, p)})} = {(f, p)}"
```

by (rule set\_eqI) auto

qed

thus "{(σ, ζ) | σ ζ s. s ∈ init (paadv i)

∧ (σ i, ζ) = id s

∧ (∀j. j ≠ i → σ j ∈ (fst o id) ' init (paadv j)) } ⊆ init (opaadv i)"

by simp

next

show "∀j. init (paadv j) ≠ {}"

unfolding σ<sub>AADV</sub>\_def by simp

next

fix i s a s' σ σ'

assume "σ i = fst (id s)"

and "σ' i = fst (id s)'"

and "(s, a, s') ∈ trans (paadv i)"

then obtain q q' where "s = (σ i, q)"

and "s' = (σ' i, q)'"

and "((σ i, q), a, (σ' i, q')) ∈ trans (paadv i)"

by (cases s, cases s') auto

```

from this(3) have "((σ, q), a, (σ', q')) ∈ trans (opaadv i)"
  by simp (rule open_seqp_action [OF aadv_wf])

with ⟨s = (σ i, q)⟩ and ⟨s' = (σ' i, q')⟩
  show "((σ, snd (id s)), a, (σ', snd (id s'))) ∈ trans (opaadv i)"
    by simp
qed
then interpret op: openproc paadv opaadv id .
have [simp]: "∧i. (SOME x. x ∈ (fst o id) ' init (paadv i)) = aadv_init i"
  unfolding σ_AODV_def by simp
hence "∧i. openproc.initmissing paadv id i = initmissing i"
  unfolding op.initmissing_def op.someinit_def initmissing_def
  by (auto split: option.split)
thus "openproc.initmissing paadv id = initmissing" ..
qed

interpretation aadv_openproc_par_qmsg: openproc_parq paadv opaadv id qmsg
where "aadv_openproc_par_qmsg.netglobal = netglobal"
  and "aadv_openproc_par_qmsg.initmissing = initmissing"
proof -
  show "openproc_parq paadv opaadv id qmsg"
    by (unfold_locales) simp
  then interpret opq: openproc_parq paadv opaadv id qmsg .

  have im: "∧σ. openproc.initmissing (λi. paadv i ⟨⟨ qmsg⟩⟩ (λ(p, q). (fst (id p), snd (id p), q))) σ
    = initmissing σ"
    unfolding opq.initmissing_def opq.someinit_def initmissing_def
    unfolding σ_AODV_def σ_QMSG_def by (clarsimp cong: option.case_cong)
  thus "openproc.initmissing (λi. paadv i ⟨⟨ qmsg⟩⟩ (λ(p, q). (fst (id p), snd (id p), q))) = initmissing"
    by (rule ext)
  have "∧P σ. openproc.netglobal (λi. paadv i ⟨⟨ qmsg⟩⟩ (λ(p, q). (fst (id p), snd (id p), q))) P σ
    = netglobal P σ"
    unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def
    unfolding σ_AODV_def σ_QMSG_def
    by (clarsimp cong: option.case_cong
      simp del: One_nat_def
      simp add: fst_initmissing_netgmap_default_aadv_init_netlift
        [symmetric, unfolded initmissing_def])
  thus "openproc.netglobal (λi. paadv i ⟨⟨ qmsg⟩⟩ (λ(p, q). (fst (id p), snd (id p), q))) = netglobal"
    by auto
qed

lemma net_nhop_quality_increases:
  assumes "wf_net_tree n"
  shows "closed (pnet (λi. paadv i ⟨⟨ qmsg⟩⟩ n) |||= netglobal
    (λσ. ∀i dip. let nhip = the (nhop (rt (σ i)) dip)
      in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
      → (rt (σ i)) ⊆dip (rt (σ nhip))))"
    (is "_ |||= netglobal (λσ. ∀i. ?inv σ i)")
proof -
  from ⟨wf_net_tree n⟩
  have proto: "closed (pnet (λi. paadv i ⟨⟨ qmsg⟩⟩ n) |||= netglobal (λσ. ∀i∈net_tree_ips n. ∀dip.
    let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊆dip (rt (σ nhip))))"
    by (rule aadv_openproc_par_qmsg.close_opnet [OF _ onet_nhop_quality_increases])
  show ?thesis
  unfolding invariant_def opnet_sos.opnet_tau1
  proof (rule, simp only: aadv_openproc_par_qmsg.netglobalsimp
    fst_initmissing_netgmap_pair_fst, rule allI)
    fix σ i
    assume sr: "σ ∈ reachable (closed (pnet (λi. paadv i ⟨⟨ qmsg⟩⟩ n)) TT)"
    hence "∀i∈net_tree_ips n. ?inv (fst (initmissing (netgmap fst σ))) i"
      by - (drule invariantD [OF proto],

```

```

      simp only: aadv_openproc_par_qmsg.netglobalsimp
                fst_initmissing_netgmap_pair_fst)
thus "?inv (fst (initmissing (netgmap fst  $\sigma$ ))) i"
proof (cases "i $\in$ net_tree_ips n")
  assume "i $\notin$ net_tree_ips n"
  from sr have " $\sigma \in$  reachable (pnet ( $\lambda i$ . paadv i  $\langle\langle$  qmsg) n) TT" ..
  hence "net_ips  $\sigma =$  net_tree_ips n" ..
  with (i $\notin$ net_tree_ips n) have "i $\notin$ net_ips  $\sigma$ " by simp
  hence "(fst (initmissing (netgmap fst  $\sigma$ ))) i = aadv_init i"
    by simp
  thus ?thesis by simp
qed metis
qed
qed

```

### 3.11.6 Loop freedom of AODV

**theorem** aadv\_loop\_freedom:

assumes "wf\_net\_tree n"

shows "closed (pnet ( $\lambda i$ . paadv i  $\langle\langle$  qmsg) n)  $\models$  netglobal ( $\lambda \sigma$ .  $\forall dip$ . irrefl ((rt\_graph  $\sigma$  dip)<sup>+</sup>))"

using assms by (rule aadv\_openproc\_par\_qmsg.netglobal\_weakenE

[OF net\_nhop\_quality\_increases\_inv\_to\_loop\_freedom])

end

# Chapter 4

## Variant D: Forwarding the Route Request

Explanation [4, §10.5]: In AODV's route discovery process, a destination node (or an intermediate node with an active route to the destination) will generate a RREP message in response to a received RREQ message. The RREQ message is then dropped and not forwarded. This termination of the route discovery process at the destination can lead to other nodes inadvertently creating non-optimal routes to the source node [5]. A possible modification to solve this problem is to allow the destination node to continue to forward the RREQ message. A route request is only stopped if it has been handled before. The forwarded RREQ message from the destination node needs to be modified to include a Boolean flag `handled` that indicates a RREP message has already been generated and sent in response to the former message. In case the flag is set to true, it prevents other nodes (with valid route to the destination) from sending a RREP message in response to their reception of the forwarded RREQ message.

### 4.1 Predicates and functions used in the AODV model

```
theory D_Aodv_Data
imports D_Fwdrreqs
begin
```

#### 4.1.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

```
definition inc :: "sqn  $\Rightarrow$  sqn"
  where "inc sn  $\equiv$  if sn = 0 then sn else sn + 1"
```

```
lemma less_than_inc [simp]: "x  $\leq$  inc x"
  unfolding inc_def by simp
```

```
lemma inc_minus_suc_0 [simp]:
  "inc x - Suc 0 = x"
  unfolding inc_def by simp
```

```
lemma inc_never_one' [simp, intro]: "inc x  $\neq$  Suc 0"
  unfolding inc_def by simp
```

```
lemma inc_never_one [simp, intro]: "inc x  $\neq$  1"
  by simp
```

#### 4.1.2 Modelling Routes

A route is a 6-tuple,  $(dsn, dsk, flag, hops, nhop, pre)$  where  $dsn$  is the 'destination sequence number',  $dsk$  is the 'destination-sequence-number status',  $flag$  is the route status,  $hops$  is the number of hops to the destination,  $nhop$  is the next hop toward the destination, and  $pre$  is the set of 'precursor nodes' those interested in hearing about changes to the route.

```
type_synonym r = "sqn  $\times$  k  $\times$  f  $\times$  nat  $\times$  ip  $\times$  ip set"
```

```
definition proj2 :: "r  $\Rightarrow$  sqn" (" $\pi_2$ ")
```

```

where "π2 ≡ λ(dsn, _, _, _, _). dsn"

definition proj3 :: "r ⇒ k" ("π3")
  where "π3 ≡ λ(_, dsk, _, _, _). dsk"

definition proj4 :: "r ⇒ f" ("π4")
  where "π4 ≡ λ(_, _, flag, _, _). flag"

definition proj5 :: "r ⇒ nat" ("π5")
  where "π5 ≡ λ(_, _, _, hops, _, _). hops"

definition proj6 :: "r ⇒ ip" ("π6")
  where "π6 ≡ λ(_, _, _, _, nhip, _). nhip"

definition proj7 :: "r ⇒ ip set" ("π7")
  where "π7 ≡ λ(_, _, _, _, _, pre). pre"

lemma projs [simp]:
  "π2(dsn, dsk, flag, hops, nhip, pre) = dsn"
  "π3(dsn, dsk, flag, hops, nhip, pre) = dsk"
  "π4(dsn, dsk, flag, hops, nhip, pre) = flag"
  "π5(dsn, dsk, flag, hops, nhip, pre) = hops"
  "π6(dsn, dsk, flag, hops, nhip, pre) = nhip"
  "π7(dsn, dsk, flag, hops, nhip, pre) = pre"
  by (clarsimp simp: proj2_def proj3_def proj4_def
      proj5_def proj6_def proj7_def)+

lemma proj3_pred [intro]: "[[ P kno; P unk ] ⇒ P (π3 x)]"
  by (rule k.induct)

lemma proj4_pred [intro]: "[[ P val; P inv ] ⇒ P (π4 x)]"
  by (rule f.induct)

lemma proj6_pair_snd [simp]:
  fixes dsn' r
  shows "π6 (dsn', snd (r)) = π6(r)"
  by (cases r) simp



### 4.1.3 Routing Tables



Routing tables map ip addresses to route entries.



```

type_synonym rt = "ip → r"

syntax
  "_Sigma_route" :: "rt ⇒ ip → r" ("σroute'(_, _)'")

translations
  "σroute(rt, dip)" => "rt dip"

definition sqn :: "rt ⇒ ip ⇒ sqn"
  where "sqn rt dip ≡ case σroute(rt, dip) of Some r ⇒ π2(r) | None ⇒ 0"

definition sqnf :: "rt ⇒ ip ⇒ k"
  where "sqnf rt dip ≡ case σroute(rt, dip) of Some r ⇒ π3(r) | None ⇒ unk"

abbreviation flag :: "rt ⇒ ip → f"
  where "flag rt dip ≡ map_option π4 (σroute(rt, dip))"

abbreviation dhops :: "rt ⇒ ip → nat"
  where "dhops rt dip ≡ map_option π5 (σroute(rt, dip))"

abbreviation nhop :: "rt ⇒ ip → ip"
  where "nhop rt dip ≡ map_option π6 (σroute(rt, dip))"

```


```



```

abbreviation precS :: "rt  $\Rightarrow$  ip  $\rightarrow$  ip set"
  where "precS rt dip  $\equiv$  map_option  $\pi_7$  ( $\sigma_{route}(rt, dip)$ )"

definition vD :: "rt  $\Rightarrow$  ip set"
  where "vD rt  $\equiv$  {dip. flag rt dip = Some val}"

definition iD :: "rt  $\Rightarrow$  ip set"
  where "iD rt  $\equiv$  {dip. flag rt dip = Some inv}"

definition kD :: "rt  $\Rightarrow$  ip set"
  where "kD rt  $\equiv$  {dip. rt dip  $\neq$  None}"

lemma kD_is_vD_and_iD: "kD rt = vD rt  $\cup$  iD rt"
  unfolding kD_def vD_def iD_def by auto

lemma vD_iD_gives_kD [simp]:
  " $\bigwedge$  ip rt. ip  $\in$  vD rt  $\implies$  ip  $\in$  kD rt"
  " $\bigwedge$  ip rt. ip  $\in$  iD rt  $\implies$  ip  $\in$  kD rt"
  unfolding kD_is_vD_and_iD by simp_all

lemma kD_Some [dest]:
  fixes dip rt
  assumes "dip  $\in$  kD rt"
  shows " $\exists$  dsn dsk flag hops nhip pre.
     $\sigma_{route}(rt, dip) = \text{Some} (dsn, dsk, flag, hops, nhip, pre)$ "
  using assms unfolding kD_def by simp

lemma kD_None [dest]:
  fixes dip rt
  assumes "dip  $\notin$  kD rt"
  shows " $\sigma_{route}(rt, dip) = \text{None}$ "
  using assms unfolding kD_def
  by (metis (mono_tags) mem_Collect_eq)

lemma vD_Some [dest]:
  fixes dip rt
  assumes "dip  $\in$  vD rt"
  shows " $\exists$  dsn dsk hops nhip pre.
     $\sigma_{route}(rt, dip) = \text{Some} (dsn, dsk, val, hops, nhip, pre)$ "
  using assms unfolding vD_def by simp

lemma vD_empty [simp]: "vD Map.empty = {}"
  unfolding vD_def by simp

lemma iD_Some [dest]:
  fixes dip rt
  assumes "dip  $\in$  iD rt"
  shows " $\exists$  dsn dsk hops nhip pre.
     $\sigma_{route}(rt, dip) = \text{Some} (dsn, dsk, inv, hops, nhip, pre)$ "
  using assms unfolding iD_def by simp

lemma val_is_vD [elim]:
  fixes ip rt
  assumes "ip  $\in$  kD(rt)"
  and "the (flag rt ip) = val"
  shows "ip  $\in$  vD(rt)"
  using assms unfolding vD_def by auto

lemma inv_is_iD [elim]:
  fixes ip rt
  assumes "ip  $\in$  kD(rt)"
  and "the (flag rt ip) = inv"
  shows "ip  $\in$  iD(rt)"
  using assms unfolding iD_def by auto

```

```

lemma iD_flag_is_inv [elim, simp]:
  fixes ip rt
  assumes "ip ∈ iD(rt)"
  shows "the (flag rt ip) = inv"
proof -
  from ⟨ip ∈ iD(rt)⟩ have "ip ∈ kD(rt)" by auto
  with assms show ?thesis unfolding iD_def by auto
qed

lemma kD_but_not_vD_is_iD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "ip ∉ vD(rt)"
  shows "ip ∈ iD(rt)"
proof -
  from ⟨ip ∈ kD(rt)⟩ obtain dsn dsk f hops nhop pre
  where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop, pre)"
  by (metis kD_Some)
  from ⟨ip ∉ vD(rt)⟩ have "f ≠ val"
  proof (rule contrapos_nn)
    assume "f = val"
    with rtip have "the (flag rt ip) = val" by simp
    with ⟨ip ∈ kD(rt)⟩ show "ip ∈ vD(rt)" ..
  qed
  with rtip have "the (flag rt ip) = inv" by simp
  with ⟨ip ∈ kD(rt)⟩ show "ip ∈ iD(rt)" ..
qed

lemma vD_or_iD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "ip ∈ vD(rt) ⇒ P rt ip"
  and "ip ∈ iD(rt) ⇒ P rt ip"
  shows "P rt ip"
proof -
  from ⟨ip ∈ kD(rt)⟩ have "ip ∈ vD(rt) ∪ iD(rt)"
  by (simp add: kD_is_vD_and_iD)
  thus ?thesis by (auto elim: assms(2-3))
qed

lemma proj5_eq_dhops: "∧ dip rt. dip ∈ kD(rt) ⇒ π5(the (rt dip)) = the (dhops rt dip)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj4_eq_flag: "∧ dip rt. dip ∈ kD(rt) ⇒ π4(the (rt dip)) = the (flag rt dip)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj2_eq_sqn: "∧ dip rt. dip ∈ kD(rt) ⇒ π2(the (rt dip)) = sqn rt dip"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 [simp]:
  "∧ ip rt. ip ∈ kD(rt) ⇒ sqnf rt ip = π3(the (rt ip))"
unfolding sqnf_def by auto

lemma vD_flag_val [simp]:
  "∧ dip rt. dip ∈ vD (rt) ⇒ the (flag rt dip) = val"
unfolding vD_def by clarsimp

lemma kD_update [simp]:
  "∧ rt nip v. kD (rt(nip ↦ v)) = insert nip (kD rt)"
unfolding kD_def by auto

lemma kD_empty [simp]: "kD Map.empty = {}"
unfolding kD_def by simp

```

```

lemma ip_equal_or_known [elim]:
  fixes rt ip ip'
  assumes "ip = ip'  $\vee$  ip  $\in$  kD(rt)"
  and "ip = ip'  $\implies$  P rt ip ip'"
  and "[[ ip  $\neq$  ip'; ip  $\in$  kD(rt)]]  $\implies$  P rt ip ip'"
  shows "P rt ip ip'"
  using assms by auto

```

#### 4.1.4 Updating Routing Tables

Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.

##### Updating Precursor Lists

```

definition addpre :: "r  $\Rightarrow$  ip set  $\Rightarrow$  r"
  where "addpre r npre  $\equiv$  let (dsn, dsk, flag, hops, nhip, pre) = r in
        (dsn, dsk, flag, hops, nhip, pre  $\cup$  npre)"

```

```

lemma proj2_addpre:
  fixes v pre
  shows " $\pi_2$ (addpre v pre) =  $\pi_2$ (v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj3_addpre:
  fixes v pre
  shows " $\pi_3$ (addpre v pre) =  $\pi_3$ (v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj4_addpre:
  fixes v pre
  shows " $\pi_4$ (addpre v pre) =  $\pi_4$ (v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj5_addpre:
  fixes v pre
  shows " $\pi_5$ (addpre v pre) =  $\pi_5$ (v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj6_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows " $\pi_6$ (addpre v npre) =  $\pi_6$ (v)"
  unfolding addpre_def by (cases v) simp

```

```

lemma proj7_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows " $\pi_7$ (addpre v npre) =  $\pi_7$ (v)  $\cup$  npre"
  unfolding addpre_def by (cases v) simp

```

```

lemma addpre_empty: "addpre r {} = r"
  unfolding addpre_def by simp

```

```

lemma addpre_r:
  "addpre (dsn, dsk, fl, hops, nhip, pre) npre = (dsn, dsk, fl, hops, nhip, pre  $\cup$  npre)"
  unfolding addpre_def by simp

```

```

lemmas addpre_simps [simp] = proj2_addpre proj3_addpre proj4_addpre proj5_addpre
  proj6_addpre proj7_addpre addpre_empty addpre_r

```

```

definition addpreRT :: "rt  $\Rightarrow$  ip  $\Rightarrow$  ip set  $\rightarrow$  rt"
  where "addpreRT rt dip npre  $\equiv$ 
        map_option ( $\lambda$ s. rt (dip  $\mapsto$  addpre s npre)) ( $\sigma_{route}(rt, dip)$ )"

```

```

lemma snd_addpre [simp]:
  " $\bigwedge$  dsn dsn' v pre. (dsn, snd(addpre (dsn', v) pre)) = addpre (dsn, v) pre"
  unfolding addpre_def by clarsimp

lemma proj2_addpreRT [simp]:
  fixes ip rt ip' npre
  assumes "ip  $\in$  kD rt"
  and "ip'  $\in$  kD rt"
  shows " $\pi_2$ (the (the (addpreRT rt ip' npre) ip)) =  $\pi_2$ (the (rt ip))"
  using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj3_addpreRT [simp]:
  fixes ip rt ip' npre
  assumes "ip  $\in$  kD rt"
  and "ip'  $\in$  kD rt"
  shows " $\pi_3$ (the (the (addpreRT rt ip' npre) ip)) =  $\pi_3$ (the (rt ip))"
  using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

lemma proj5_addpreRT [simp]:
  " $\bigwedge$  rt dip ip npre. dip  $\in$  kD(rt)  $\implies$   $\pi_5$ (the (the (addpreRT rt dip npre) ip)) =  $\pi_5$ (the (rt ip))"
  unfolding addpreRT_def by auto

lemma flag_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip  $\in$  kD rt"
  shows "flag (the (addpreRT rt dip pre)) ip = flag rt ip"
  unfolding addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

lemma kD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip  $\in$  kD rt"
  shows "kD (the (addpreRT rt dip npre)) = kD rt"
  unfolding kD_def addpreRT_def
  using assms [THEN kD_Some]
  by clarsimp blast

lemma vD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip  $\in$  kD rt"
  shows "vD (the (addpreRT rt dip npre)) = vD rt"
  unfolding vD_def addpreRT_def
  using assms [THEN kD_Some] by clarsimp auto

lemma iD_addpreRT [simp]:
  fixes rt dip npre
  assumes "dip  $\in$  kD rt"
  shows "iD (the (addpreRT rt dip npre)) = iD rt"
  unfolding iD_def addpreRT_def
  using assms [THEN kD_Some] by clarsimp auto

lemma nhop_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip  $\in$  kD rt"
  shows "nhop (the (addpreRT rt dip pre)) ip = nhop rt ip"
  unfolding sqn_def addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

lemma sqn_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip  $\in$  kD rt"
  shows "sqn (the (addpreRT rt dip pre)) ip = sqn rt ip"
  unfolding sqn_def addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

```

```

lemma dhops_addpreRT [simp]:
  fixes rt pre ip dip
  assumes "dip ∈ kD rt"
  shows "dhops (the (addpreRT rt dip pre)) ip = dhops rt ip"
  unfolding addpreRT_def
  using assms [THEN kD_Some] by (clarsimp)

lemma sqnf_addpreRT [simp]:
  " $\bigwedge ip dip. ip \in kD(rt \ \xi) \implies sqnf (the (addpreRT (rt \ \xi) ip npre)) dip = sqnf (rt \ \xi) dip$ "
  unfolding sqnf_def addpreRT_def by auto

```

## Updating route entries

```

lemma in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∈ kD(rt)"
  shows "(case rt dip of None  $\Rightarrow$  en | Some r  $\Rightarrow$  es r) = es (the (rt dip))"
  using assms [THEN kD_Some] by auto

```

```

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip  $\notin$  kD(rt)"
  shows "(case rt dip of None  $\Rightarrow$  en | Some r  $\Rightarrow$  es r) = en"
  using assms [THEN kD_None] by auto

```

```

lemma rt_Some_sqn [dest]:
  fixes rt and ip dsn dsk flag hops nhip pre
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows "sqn rt ip = dsn"
  unfolding sqn_def using assms by simp

```

```

lemma not_kD_sqn [simp]:
  fixes dip rt
  assumes "dip  $\notin$  kD(rt)"
  shows "sqn rt dip = 0"
  using assms unfolding sqn_def
  by simp

```

```

definition update_arg_wf :: "r  $\Rightarrow$  bool"
where "update_arg_wf r  $\equiv$   $\pi_4(r) = \text{val} \wedge$ 
      ( $\pi_2(r) = 0$ ) = ( $\pi_3(r) = \text{unk}$ )  $\wedge$ 
      ( $\pi_3(r) = \text{unk} \longrightarrow \pi_5(r) = 1$ )"

```

```

lemma update_arg_wf_gives_cases:
  " $\bigwedge r. \text{update\_arg\_wf } r \implies (\pi_2(r) = 0) = (\pi_3(r) = \text{unk})$ "
  unfolding update_arg_wf_def by simp

```

```

lemma update_arg_wf_tuples [simp]:
  " $\bigwedge nhip \text{ pre}. \text{update\_arg\_wf } (0, \text{unk}, \text{val}, \text{Suc } 0, nhip, \text{pre})$ "
  " $\bigwedge n \text{ hops } nhip \text{ pre}. \text{update\_arg\_wf } (\text{Suc } n, \text{kno}, \text{val}, \text{hops}, nhip, \text{pre})$ "
  unfolding update_arg_wf_def by auto

```

```

lemma update_arg_wf_tuples' [elim]:
  " $\bigwedge n \text{ hops } nhip \text{ pre}. \text{Suc } 0 \leq n \implies \text{update\_arg\_wf } (n, \text{kno}, \text{val}, \text{hops}, nhip, \text{pre})$ "
  unfolding update_arg_wf_def by auto

```

```

lemma wf_r_cases [intro]:
  fixes P r
  assumes "update_arg_wf r"
  and c1: " $\bigwedge nhip \text{ pre}. P (0, \text{unk}, \text{val}, \text{Suc } 0, nhip, \text{pre})$ "
  and c2: " $\bigwedge dsn \text{ hops } nhip \text{ pre}. dsn > 0 \implies P (dsn, \text{kno}, \text{val}, \text{hops}, nhip, \text{pre})$ "
  shows "P r"
  proof -

```

```

obtain dsn dsk flag hops nhip pre
where *: "r = (dsn, dsk, flag, hops, nhip, pre)" by (cases r)
with ⟨update_arg_wf r⟩ have wf1: "flag = val"
      and wf2: "(dsn = 0) = (dsk = unk)"
      and wf3: "dsk = unk → (hops = 1)"
  unfolding update_arg_wf_def by auto
have "P (dsn, dsk, flag, hops, nhip, pre)"
proof (cases dsk)
  assume "dsk = unk"
  moreover with wf2 wf3 have "dsn = 0" and "hops = Suc 0" by auto
  ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c1)
next
  assume "dsk = kno"
  moreover with wf2 have "dsn > 0" by simp
  ultimately show ?thesis using ⟨flag = val⟩ by simp (rule c2)
qed
with * show "P r" by simp
qed

```

definition update :: "rt ⇒ ip ⇒ r ⇒ rt"

```

where
"update rt ip r ≡
  case σroute(rt, ip) of
    None ⇒ rt (ip ↦ r)
  | Some s ⇒
    if π2(s) < π2(r) then rt (ip ↦ addpre r (π7(s)))
    else if π2(s) = π2(r) ∧ (π5(s) > π5(r) ∨ π4(s) = inv)
      then rt (ip ↦ addpre r (π7(s)))
    else if π3(r) = unk
      then rt (ip ↦ (π2(s), snd (addpre r (π7(s))))
    else rt (ip ↦ addpre s (π7(r)))"

```

lemma update\_simps [simp]:

```

fixes r s nrt nr nr' ns rt ip
defines "s ≡ the σroute(rt, ip)"
  and "nr ≡ addpre r (π7(s))"
  and "nr' ≡ (π2(s), π3(nr), π4(nr), π5(nr), π6(nr), π7(nr))"
  and "ns ≡ addpre s (π7(r))"

```

shows

```

"[[ip ∉ kD(rt)] ⇒ update rt ip r = rt (ip ↦ r)"
"[[ip ∈ kD(rt); sqn rt ip < π2(r)] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); sqn rt ip = π2(r);
  the (dhops rt ip) > π5(r)] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); sqn rt ip = π2(r);
  flag rt ip = Some inv] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); π3(r) = unk; (π2(r) = 0) = (π3(r) = unk)] ⇒ update rt ip r = rt (ip ↦ nr)"
"[[ip ∈ kD(rt); sqn rt ip ≥ π2(r); π3(r) = kno;
  sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val ]
  ⇒ update rt ip r = rt (ip ↦ ns)"

```

proof -

```

  assume "ip ∉ kD(rt)"
  hence "σroute(rt, ip) = None" ..
  thus "update rt ip r = rt (ip ↦ r)"
    unfolding update_def by simp
next
  assume "ip ∈ kD(rt)"
  and "sqn rt ip < π2(r)"
  from this(1) obtain dsn dsk fl hops nhip pre
  where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with ⟨sqn rt ip < π2(r)⟩ show "update rt ip r = rt (ip ↦ nr)"
    unfolding update_def nr_def s_def by auto
next
  assume "ip ∈ kD(rt)"

```

```

    and "sqn rt ip =  $\pi_2(r)$ "
    and "the (dhops rt ip) >  $\pi_5(r)$ "
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with (sqn rt ip =  $\pi_2(r)$ ) and (the (dhops rt ip) >  $\pi_5(r)$ )
    show "update rt ip r = rt (ip  $\mapsto$  nr)"
    unfolding update_def nr_def s_def by auto
next
  assume "ip  $\in$  kD(rt)"
    and "sqn rt ip =  $\pi_2(r)$ "
    and "flag rt ip = Some inv"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with (sqn rt ip =  $\pi_2(r)$ ) and (flag rt ip = Some inv)
    show "update rt ip r = rt (ip  $\mapsto$  nr)"
    unfolding update_def nr_def s_def by auto
next
  assume "ip  $\in$  kD(rt)"
    and " $\pi_3(r) = \text{unk}$ "
    and " $(\pi_2(r) = 0) = (\pi_3(r) = \text{unk})$ "
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with (( $\pi_2(r) = 0$ ) = ( $\pi_3(r) = \text{unk}$ )) and ( $\pi_3(r) = \text{unk}$ )
    show "update rt ip r = rt (ip  $\mapsto$  nr)"
    unfolding update_def nr'_def nr_def s_def
    by (cases r) simp
next
  assume "ip  $\in$  kD(rt)"
    and otherassms: "sqn rt ip  $\geq$   $\pi_2(r)$ "
    " $\pi_3(r) = \text{kno}$ "
    "sqn rt ip =  $\pi_2(r) \implies$  the (dhops rt ip)  $\leq$   $\pi_5(r) \wedge$  the (flag rt ip) = val"
  from this(1) obtain dsn dsk fl hops nhip pre
    where "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
  with otherassms show "update rt ip r = rt (ip  $\mapsto$  ns)"
    unfolding update_def ns_def s_def by auto
qed

```

lemma update\_cases [elim]:

```

  assumes " $(\pi_2(r) = 0) = (\pi_3(r) = \text{unk})$ "
    and c1: " $\llbracket \text{ip} \notin \text{kD}(rt) \rrbracket \implies P(\text{rt } (\text{ip} \mapsto r))$ "

    and c2: " $\llbracket \text{ip} \in \text{kD}(rt); \text{sqn rt ip} < \pi_2(r) \rrbracket$ 
       $\implies P(\text{rt } (\text{ip} \mapsto \text{addpre } r (\pi_7(\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip}))))$ "
    and c3: " $\llbracket \text{ip} \in \text{kD}(rt); \text{sqn rt ip} = \pi_2(r); \text{the } (\text{dhops rt ip}) > \pi_5(r) \rrbracket$ 
       $\implies P(\text{rt } (\text{ip} \mapsto \text{addpre } r (\pi_7(\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip}))))$ "
    and c4: " $\llbracket \text{ip} \in \text{kD}(rt); \text{sqn rt ip} = \pi_2(r); \text{the } (\text{flag rt ip}) = \text{inv} \rrbracket$ 
       $\implies P(\text{rt } (\text{ip} \mapsto \text{addpre } r (\pi_7(\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip}))))$ "
    and c5: " $\llbracket \text{ip} \in \text{kD}(rt); \pi_3(r) = \text{unk} \rrbracket$ 
       $\implies P(\text{rt } (\text{ip} \mapsto (\pi_2(\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip})), \pi_3(r),$ 
         $\pi_4(r), \pi_5(r), \pi_6(r), \pi_7(\text{addpre } r (\pi_7(\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip}))))))$ "
    and c6: " $\llbracket \text{ip} \in \text{kD}(rt); \text{sqn rt ip} \geq \pi_2(r); \pi_3(r) = \text{kno};$ 
       $\text{sqn rt ip} = \pi_2(r) \implies \text{the } (\text{dhops rt ip}) \leq \pi_5(r) \wedge \text{the } (\text{flag rt ip}) = \text{val} \rrbracket$ 
       $\implies P(\text{rt } (\text{ip} \mapsto \text{addpre } (\text{the } \sigma_{\text{route}}(\text{rt}, \text{ip})) (\pi_7(r))))$ "
  shows "(P (update rt ip r))"
  proof (cases "ip  $\in$  kD(rt)")
    assume "ip  $\notin$  kD(rt)"
    with c1 show ?thesis
      by simp
  next
    assume "ip  $\in$  kD(rt)"

```

```

moreover then obtain dsn dsk fl hops nhip pre
  where rteq: "rt ip = Some (dsn, dsk, fl, hops, nhip, pre)"
    by (metis kD_Some)
moreover obtain dsn' dsk' fl' hops' nhip' pre'
  where req: "r = (dsn', dsk', fl', hops', nhip', pre')"
    by (cases r) metis
ultimately show ?thesis
  using ⟨(π2(r) = 0) = (π3(r) = unk)⟩
    c2 [OF ⟨ip ∈ kD(rt)⟩]
    c3 [OF ⟨ip ∈ kD(rt)⟩]
    c4 [OF ⟨ip ∈ kD(rt)⟩]
    c5 [OF ⟨ip ∈ kD(rt)⟩]
    c6 [OF ⟨ip ∈ kD(rt)⟩]
  unfolding update_def sqn_def by auto
qed

```

lemma update\_cases\_kD:

```

assumes "(π2(r) = 0) = (π3(r) = unk)"
  and "ip ∈ kD(rt)"
  and c2: "sqn rt ip < π2(r) ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip)))))"
  and c3: "[[sqn rt ip = π2(r); the (dhops rt ip) > π5(r)]
    ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip))))]"
  and c4: "[[sqn rt ip = π2(r); the (flag rt ip) = inv]
    ⇒ P (rt (ip ↦ addpre r (π7(the σroute(rt, ip))))]"
  and c5: "π3(r) = unk ⇒ P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r),
    π4(r), π5(r), π6(r),
    π7(addpre r (π7(the σroute(rt, ip))))))"
  and c6: "[[sqn rt ip ≥ π2(r); π3(r) = kno;
    sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val]
    ⇒ P (rt (ip ↦ addpre (the σroute(rt, ip)) (π7(r))))"
shows "(P (update rt ip r))"
using assms(1) proof (rule update_cases)
  assume "sqn rt ip < π2(r)"
  thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))" by (rule c2)
next
  assume "sqn rt ip = π2(r)"
  and "the (dhops rt ip) > π5(r)"
  thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))"
  by (rule c3)
next
  assume "sqn rt ip = π2(r)"
  and "the (flag rt ip) = inv"
  thus "P (rt(ip ↦ addpre r (π7(the (rt ip)))))"
  by (rule c4)
next
  assume "π3(r) = unk"
  thus "P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r), π4(r), π5(r), π6(r),
    π7(addpre r (π7(the (rt ip))))))"
  by (rule c5)
next
  assume "sqn rt ip ≥ π2(r)"
  and "π3(r) = kno"
  and "sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val"
  thus "P (rt (ip ↦ addpre (the (rt ip)) (π7(r))))"
  by (rule c6)
qed (simp add: ⟨ip ∈ kD(rt)⟩)

```

lemma in\_kD\_after\_update [simp]:

```

fixes rt nip dsn dsk flag hops nhip pre
shows "kD (update rt nip (dsn, dsk, flag, hops, nhip, pre)) = insert nip (kD rt)"
unfolding update_def
by (cases "rt nip") auto

```

lemma nhop\_of\_update [simp]:



```

fixes rt dip dsn dsk flag hops nhip
assumes "rt  $\neq$  update rt dip (dsn, dsk, flag, hops, nhip, {})"
shows "the (nhop (update rt dip (dsn, dsk, flag, hops, nhip, {})) dip) = nhip"
proof -
from assms
have update_neq: " $\bigwedge v. rt\ dip = Some\ v \implies$ 
  update rt dip (dsn, dsk, flag, hops, nhip, {})
   $\neq$  rt(dip  $\mapsto$  addpre (the (rt dip)) ( $\pi_7$  (dsn, dsk, flag, hops, nhip, {})))"
  by auto
show ?thesis
proof (cases "rt dip = None")
  assume "rt dip = None"
  thus "?thesis" unfolding update_def by clarsimp
next
  assume "rt dip  $\neq$  None"
  then obtain v where "rt dip = Some v" by (metis not_None_eq)
  with update_neq [OF this] show ?thesis
    unfolding update_def by auto
qed
qed

lemma sqn_if_updated:
fixes rip v rt ip
shows "sqn ( $\lambda x. if\ x = rip\ then\ Some\ v\ else\ rt\ x$ ) ip
  = (if ip = rip then  $\pi_2(v)$  else sqn rt ip)"
unfolding sqn_def by simp

lemma update_sqn [simp]:
fixes rt dip rip dsn dsk hops nhip pre
assumes "(dsn = 0) = (dsk = unk)"
shows "sqn rt dip  $\leq$  sqn (update rt rip (dsn, dsk, val, hops, nhip, pre)) dip"
proof (rule update_cases)
  show "( $\pi_2$  (dsn, dsk, val, hops, nhip, pre) = 0) = ( $\pi_3$  (dsn, dsk, val, hops, nhip, pre) = unk)"
  by simp (rule assms)
qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
fixes rt ip ip' dsn dsk flag hops nhip pre
assumes "1  $\leq$  hops"
shows "sqn rt ip  $\leq$  sqn (update rt ip' (dsn, dsk, flag, hops, nhip, pre)) ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
fixes rt dsn dsk flag hops ip rip nhip pre
assumes ex: " $\forall ip \in kD\ rt. the\ (dhops\ rt\ ip) \geq 1$ "
  and ip: "(ip = rip  $\wedge$  Suc 0  $\leq$  hops)  $\vee$  (ip  $\neq$  rip  $\wedge$  ip  $\in kD\ rt)"
shows "Suc 0  $\leq$  the (dhops (update rt rip (dsn, dsk, flag, hops, nhip, pre)) ip)"
using ip proof
  assume "ip = rip  $\wedge$  Suc 0  $\leq$  hops" thus ?thesis
    unfolding update_def using ex
    by (cases "rip  $\in kD\ rt$ ") (drule(1) bspec, auto)
next
  assume "ip  $\neq$  rip  $\wedge$  ip  $\in kD\ rt$ " thus ?thesis
    using ex unfolding update_def
    by (cases "rip  $\in kD\ rt$ ") auto
qed

lemma update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip pre
assumes "ip  $\neq$  dip"
shows "(update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = rt ip"
using assms unfolding update_def
by (clarsimp split: option.split)$ 
```

```

lemma nhop_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = nhop rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma dhops_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = dhops rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
  " $\bigwedge$ rt ip dsn dsk flag hops nhip pre. sqn (rt(ip  $\mapsto$  v)) ip =  $\pi_2$ (v)"
  unfolding sqn_def by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip
  assumes "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nhip, {})) dip) = hops"
  using assms unfolding update_def
  by (clarsimp split: option.split_asm option.split split_if_asm) auto

lemma nhop_update_unk_val [simp]:
  " $\bigwedge$ rt dip ip dsn hops npre.
  the (nhop (update rt dip (dsn, unk, val, hops, ip, npre)) dip) = ip"
  unfolding update_def by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
  assumes "update rt dip (dsn, dsk, flg, hops, sip, {}) ≠ rt"
  shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip, {})) dip) = sip"
  using assms unfolding update_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma update_rt_split_asm:
  " $\bigwedge$ rt ip dsn dsk flag hops sip.
  P (update rt ip (dsn, dsk, flag, hops, sip, {}))
  =
  ( $\neg$ (rt = update rt ip (dsn, dsk, flag, hops, sip, {}))  $\wedge$   $\neg$ P rt
   $\vee$  rt ≠ update rt ip (dsn, dsk, flag, hops, sip, {})
   $\wedge$   $\neg$ P (update rt ip (dsn, dsk, flag, hops, sip, {})))"
  by auto

lemma sqn_update [simp]: " $\bigwedge$ rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
   $\implies$  sqn (update rt dip (dsn, kno, flg, hops, sip, {})) dip = dsn"
  unfolding update_def by (clarsimp split: option.split split_if_asm) auto

lemma sqnf_update [simp]: " $\bigwedge$ rt dip dsn dsk flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
   $\implies$  sqnf (update rt dip (dsn, dsk, flg, hops, sip, {})) dip = dsk"
  unfolding update_def sqnf_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma update_kno_dsn_greater_zero:
  " $\bigwedge$ rt dip ip dsn hops npre. 1 ≤ dsn  $\implies$  1 ≤ (sqn (update rt dip (dsn, kno, val, hops, ip, npre)) dip)"
  unfolding update_def
  by (clarsimp split: option.splits)

lemma proj3_update [simp]: " $\bigwedge$ rt dip dsn dsk flg hops sip.

```

```

rt ≠ update rt dip (dsn, dsk, flg, hops, sip, { })
⇒ π3(the (update rt dip (dsn, dsk, flg, hops, sip, { }) dip)) = dsk"
unfolding update_def sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

lemma nhop_update_changed_kno_val [simp]: "∧rt ip dsn dsk hops nhip.
rt ≠ update rt ip (dsn, kno, val, hops, nhip, { })
⇒ the (nhop (update rt ip (dsn, kno, val, hops, nhip, { })) ip) = nhip"
using assms unfolding update_def
by (clarsimp split: option.split_asm option.split split_if_asm) auto

lemma flag_update [simp]: "∧rt dip dsn flg hops sip.
rt ≠ update rt dip (dsn, kno, flg, hops, sip, { })
⇒ the (flag (update rt dip (dsn, kno, flg, hops, sip, { })) dip) = flg"
unfolding update_def
by (clarsimp split: option.split split_if_asm) auto

lemma the_flag_Some [dest!]:
  fixes ip rt
  assumes "the (flag rt ip) = x"
    and "ip ∈ kD rt"
  shows "flag rt ip = Some x"
using assms by auto

lemma kD_update_unchanged [dest]:
  fixes rt dip dsn dsk flag hops nhip pre
  assumes "rt = update rt dip (dsn, dsk, flag, hops, nhip, pre)"
  shows "dip ∈ kD(rt)"
proof -
  have "dip ∈ kD(update rt dip (dsn, dsk, flag, hops, nhip, pre))" by simp
  with assms show ?thesis by simp
qed

lemma nhop_update [simp]: "∧rt dip dsn dsk flg hops sip.
rt ≠ update rt dip (dsn, dsk, flg, hops, sip, { })
⇒ the (nhop (update rt dip (dsn, dsk, flg, hops, sip, { })) dip) = sip"
unfolding update_def sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

lemma sqn_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqn rt ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.splits) auto

lemma sqnf_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip pre
  assumes "ip ≠ dip"
  shows "sqnf (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqnf rt ip"
using assms unfolding update_def sqnf_def
by (clarsimp split: option.splits) auto

lemma vD_update_val [dest]:
"∧dip rt dip' dsn dsk hops nhip pre.
dip ∈ vD(update rt dip' (dsn, dsk, val, hops, nhip, pre)) ⇒ (dip ∈ vD(rt) ∨ dip=dip'"
unfolding update_def vD_def by (clarsimp split: option.split_asm split_if_asm)

```

## Invalidating route entries

```

definition invalidate :: "rt ⇒ (ip ⇒ sqn) ⇒ rt"
where "invalidate rt dests ≡
λip. case (rt ip, dests ip) of
  (None, _) ⇒ None

```

```

| (Some s, None) ⇒ Some s
| (Some (_, dsk, _, hops, nhip, pre), Some rsn) ⇒
  Some (rsn, dsk, inv, hops, nhip, pre)"

lemma proj3_invalidate [simp]:
  "∧dip. π3(the ((invalidate rt dests) dip)) = π3(the (rt dip))"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma proj5_invalidate [simp]:
  "∧dip. π5(the ((invalidate rt dests) dip)) = π5(the (rt dip))"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma proj6_invalidate [simp]:
  "∧dip. π6(the ((invalidate rt dests) dip)) = π6(the (rt dip))"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma proj7_invalidate [simp]:
  "∧dip. π7(the ((invalidate rt dests) dip)) = π7(the (rt dip))"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_kD_inv [simp]:
  "∧rt dests. kD (invalidate rt dests) = kD rt"
  unfolding invalidate_def kD_def
  by (simp split: option.split)

lemma invalidate_sqn:
  fixes rt dip dests
  assumes "∀rsn. dests dip = Some rsn → sqn rt dip ≤ rsn"
  shows "sqn rt dip ≤ sqn (invalidate rt dests) dip"
  proof (cases "dip ∈ kD(rt)")
    assume "¬ dip ∈ kD(rt)"
    hence "dip ∈ kD(rt)" by simp
    then obtain dsn dsk flag hops nhip pre where "rt dip = Some (dsn, dsk, flag, hops, nhip, pre)"
      by (metis kD_Some)
    with assms show "sqn rt dip ≤ sqn (invalidate rt dests) dip"
      by (cases "dests dip") (auto simp add: invalidate_def sqn_def)
  qed simp

lemma sqn_invalidate_in_dests [simp]:
  fixes dests ipa rsn rt
  assumes "dests ipa = Some rsn"
    and "ipa ∈ kD(rt)"
  shows "sqn (invalidate rt dests) ipa = rsn"
  unfolding invalidate_def sqn_def
  using assms(1) assms(2) [THEN kD_Some]
  by clarsimp

lemma dhops_invalidate [simp]:
  "∧dip. the (dhops (invalidate rt dests) dip) = the (dhops rt dip)"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma sqnf_invalidate [simp]:
  "∧dip. sqnf (invalidate (rt ξ) (dests ξ)) dip = sqnf (rt ξ) dip"
  unfolding sqnf_def invalidate_def by (clarsimp split: option.split)

lemma nhop_invalidate [simp]:
  "∧dip. the (nhop (invalidate (rt ξ) (dests ξ)) dip) = the (nhop (rt ξ) dip)"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
  fixes rt dests dip
  assumes "dip ∉ dom(dests)"
  shows "invalidate rt dests dip = rt dip"
  using assms unfolding invalidate_def

```

```

by (clarsimp split: option.split_asm)

lemma invalidate_none [simp]:
  fixes rt dests dip
  assumes "dip ∉ kD(rt)"
  shows "invalidate rt dests dip = None"
  using assms unfolding invalidate_def by clarsimp

lemma vD_invalidate_vD_not_dests:
  "∧ dip rt dests. dip ∈ vD(invalidate rt dests) ⇒ dip ∈ vD(rt) ∧ dests dip = None"
  unfolding invalidate_def vD_def
  by (clarsimp split: option.split_asm)

lemma sqn_invalidate_not_in_dests [simp]:
  fixes dests dip rt
  assumes "dip ∉ dom(dests)"
  shows "sqn (invalidate rt dests) dip = sqn rt dip"
  using assms unfolding sqn_def by simp

lemma invalidate_changes:
  fixes rt dests dip dsn dsk flag hops nhip pre
  assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip, pre)"
  shows " dsn = (case dests dip of None ⇒ π2(the (rt dip)) | Some rsn ⇒ rsn)
    ∧ dsk = π3(the (rt dip))
    ∧ flag = (if dests dip = None then π4(the (rt dip)) else inv)
    ∧ hops = π5(the (rt dip))
    ∧ nhip = π6(the (rt dip))
    ∧ pre = π7(the (rt dip))"
  using assms unfolding invalidate_def
  by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: "∧ dip rt dests. dip ∈ kD (rt)
  ⇒ π3(the (invalidate rt dests dip)) = π3(the (rt dip))"
  by (clarsimp simp: invalidate_def kD_def split: option.split)

lemma dests_iD_invalidate [simp]:
  assumes "dests ip = Some rsn"
  and "ip ∈ kD(rt)"
  shows "ip ∈ iD(invalidate rt dests)"
  using assms(1) assms(2) [THEN kD_Some] unfolding invalidate_def iD_def
  by (clarsimp split: option.split)

```

#### 4.1.5 Route Requests

Generate a fresh route request identifier.

```

definition nrreqid :: "(ip × rreqid) set ⇒ ip ⇒ rreqid"
  where "nrreqid rreqs ip ≡ Max ({n. (ip, n) ∈ rreqs} ∪ {0}) + 1"

```

#### 4.1.6 Queued Packets

Functions for sending data packets.

```

type_synonym store = "ip → (p × data list)"

```

```

definition sigma_queue :: "store ⇒ ip ⇒ data list" ("σqueue'(_, _)'")
  where "σqueue(store, dip) ≡ case store dip of None ⇒ [] | Some (p, q) ⇒ q"

```

```

definition qD :: "store ⇒ ip set"
  where "qD ≡ dom"

```

```

definition add :: "data ⇒ ip ⇒ store ⇒ store"
  where "add d dip store ≡ case store dip of
    None ⇒ store (dip ↦ (req, [d]))

```

| Some (p, q) ⇒ store (dip ↦ (p, q @ [d]))"

```
lemma qD_add [simp]:
  fixes d dip store
  shows "qD(add d dip store) = insert dip (qD store)"
  unfolding add_def Let_def qD_def
  by (clarsimp split: option.split)
```

```
definition drop :: "ip ⇒ store → store"
  where "drop dip store ≡
    map_option (λ(p, q). if tl q = [] then store (dip := None)
                      else store (dip ↦ (p, tl q))) (store dip)"
```

```
definition sigma_p_flag :: "store ⇒ ip → p" ("σp-flag'(·, ·)")
  where "σp-flag(store, dip) ≡ map_option fst (store dip)"
```

```
definition unsetRRF :: "store ⇒ ip ⇒ store"
  where "unsetRRF store dip ≡ case store dip of
    None ⇒ store
    | Some (p, q) ⇒ store (dip ↦ (noreq, q))"
```

```
definition setRRF :: "store ⇒ (ip → sqn) ⇒ store"
  where "setRRF store dests ≡ λdip. if dests dip = None then store dip
    else map_option (λ(·, q). (req, q)) (store dip)"
```

#### 4.1.7 Comparison with the original technical report

The major differences with the AODV technical report of Fehnker et al are:

1. *nhop* is partial, thus a ‘*the*’ is needed, similarly for *dhops* and *addpreRT*.
2. *precs* is partial.
3.  $\sigma_{p\text{-flag}}(\text{store}, \text{dip})$  is partial.
4. The routing table (*rt*) is modelled as a map ( $\text{ip} \Rightarrow r \text{ option}$ ) rather than a set of 7-tuples, likewise, the *r* is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address (*dip*) is taken from the argument to the function, rather than a part of the result. Well-definedness then follows from the structure of the type and more related facts are available automatically, rather than having to be acquired through tedious proofs.
5. Similar remarks hold for the *dests* mapping passed to *invalidate*, and *store*.

end

## 4.2 AODV protocol messages

```
theory D_Aodv_Message
  imports D_Fwdrreqs
  begin
```

```
datatype msg =
  Rreq nat rreqid ip sqn k ip sqn ip bool
  | Rrep nat ip sqn ip ip
  | Rerr "ip → sqn" ip
  | Newpkt data ip
  | Pkt data ip ip
```

```
instantiation msg :: msg
begin
```

```
definition newpkt_def [simp]: "newpkt ≡ λ(d, dip). Newpkt d dip"
```

```
definition eq_newpkt_def: "eq_newpkt m ≡ case m of Newpkt d dip ⇒ True | _ ⇒ False"
```

```
instance by intro_classes (simp add: eq_newpkt_def)
```

end

The *msg* type models the different messages used within AODV. The instantiation as a *msg* is a technicality due to the special treatment of *newpkt* messages in the AWN SOS rules. This use of classes allows a clean separation of the AWN-specific definitions and these AODV-specific definitions.

```
definition rreq :: "nat × rreqid × ip × sqn × k × ip × sqn × ip × bool ⇒ msg"
  where "rreq ≡ λ(hops, rreqid, dip, dsn, dsk, oip, osn, sip, handled).
        Rreq hops rreqid dip dsn dsk oip osn sip handled"
```

```
lemma rreq_simp [simp]:
  "rreq(hops, rreqid, dip, dsn, dsk, oip, osn, sip, handled) = Rreq hops rreqid dip dsn dsk oip osn sip
  handled"
  unfolding rreq_def by simp
```

```
definition rrep :: "nat × ip × sqn × ip × ip ⇒ msg"
  where "rrep ≡ λ(hops, dip, dsn, oip, sip). Rrep hops dip dsn oip sip"
```

```
lemma rrep_simp [simp]:
  "rrep(hops, dip, dsn, oip, sip) = Rrep hops dip dsn oip sip"
  unfolding rrep_def by simp
```

```
definition rerr :: "(ip → sqn) × ip ⇒ msg"
  where "rerr ≡ λ(dests, sip). Rerr dests sip"
```

```
lemma rerr_simp [simp]:
  "rerr(dests, sip) = Rerr dests sip"
  unfolding rerr_def by simp
```

```
lemma not_eq_newpkt_rreq [simp]: "¬eq_newpkt (Rreq hops rreqid dip dsn dsk oip osn sip handled)"
  unfolding eq_newpkt_def by simp
```

```
lemma not_eq_newpkt_rrep [simp]: "¬eq_newpkt (Rrep hops dip dsn oip sip)"
  unfolding eq_newpkt_def by simp
```

```
lemma not_eq_newpkt_rerr [simp]: "¬eq_newpkt (Rerr dests sip)"
  unfolding eq_newpkt_def by simp
```

```
lemma not_eq_newpkt_pkt [simp]: "¬eq_newpkt (Pkt d dip sip)"
  unfolding eq_newpkt_def by simp
```

```
definition pkt :: "data × ip × ip ⇒ msg"
  where "pkt ≡ λ(d, dip, sip). Pkt d dip sip"
```

```
lemma pkt_simp [simp]:
  "pkt(d, dip, sip) = Pkt d dip sip"
  unfolding pkt_def by simp
```

end

## 4.3 The AODV protocol

```
theory D_Aodv
  imports D_Aodv_Data D_Aodv_Message
          "../.../AWN/AWN_SOS_Labels" "../.../AWN/AWN_Invariants"
begin
```

### 4.3.1 Data state

```
record state =
  ip      :: "ip"
  sn      :: "sqn"
  rt      :: "rt"
  rreqs   :: "(ip × rreqid) set"
```

```

store :: "store"

msg    :: "msg"
data   :: "data"
destds :: "ip  $\rightarrow$  sqn"
pre    :: "ip set"
rreqid :: "rreqid"
dip    :: "ip"
oip    :: "ip"
hops   :: "nat"
dsn    :: "sqn"
dsk    :: "k"
osn    :: "sqn"
sip    :: "ip"
handled:: "bool"

```

abbreviation aadv\_init :: "ip  $\Rightarrow$  state"

```

where "aadv_init i  $\equiv$  ( $\{$ 
  ip = i,
  sn = 1,
  rt = empty,
  rreqs = {},
  store = empty,

  msg    = (SOME x. True),
  data   = (SOME x. True),
  destds = (SOME x. True),
  pre    = (SOME x. True),
  rreqid = (SOME x. True),
  dip    = (SOME x. True),
  oip    = (SOME x. True),
  hops   = (SOME x. True),
  dsn    = (SOME x. True),
  dsk    = (SOME x. True),
  osn    = (SOME x. True),
  sip    = (SOME x. x  $\neq$  i),
  handled= (SOME x. True)
 $\}$ "

```

lemma some\_neq\_not\_eq [simp]: " $\neg((\text{SOME } x :: \text{nat. } x \neq i) = i)$ "  
 by (subst some\_eq\_ex) (metis zero\_neq\_numeral)

definition clear\_locals :: "state  $\Rightarrow$  state"

```

where "clear_locals  $\xi$  =  $\xi$  ( $\{$ 
  msg    := (SOME x. True),
  data   := (SOME x. True),
  destds := (SOME x. True),
  pre    := (SOME x. True),
  rreqid := (SOME x. True),
  dip    := (SOME x. True),
  oip    := (SOME x. True),
  hops   := (SOME x. True),
  dsn    := (SOME x. True),
  dsk    := (SOME x. True),
  osn    := (SOME x. True),
  sip    := (SOME x. x  $\neq$  ip  $\xi$ ),
  handled:= (SOME x. True)
 $\}$ "

```

lemma clear\_locals\_sip\_not\_ip [simp]: " $\neg(\text{sip } (\text{clear\_locals } \xi) = \text{ip } \xi)$ "  
 unfolding clear\_locals\_def by simp

lemma clear\_locals\_but\_not\_globals [simp]:  
 "ip (clear\_locals  $\xi$ ) = ip  $\xi$ "



```

"sn (clear_locals ξ) = sn ξ"
"rt (clear_locals ξ) = rt ξ"
"rreqs (clear_locals ξ) = rreqs ξ"
"store (clear_locals ξ) = store ξ"
unfolding clear_locals_def by auto

```

### 4.3.2 Auxilliary message handling definitions

**definition** *is\_newpkt*

```

where "is_newpkt ξ ≡ case msg ξ of
      Newpkt data' dip' ⇒ { ξ(|data := data', dip := dip'|) }
    | _ ⇒ {}"

```

**definition** *is\_pkt*

```

where "is_pkt ξ ≡ case msg ξ of
      Pkt data' dip' oip' ⇒ { ξ(|data := data', dip := dip', oip := oip'|) }
    | _ ⇒ {}"

```

**definition** *is\_rreq*

```

where "is_rreq ξ ≡ case msg ξ of
      Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' handled' ⇒
        { ξ(|hops := hops', rreqid := rreqid', dip := dip', dsn := dsn',
            dsk := dsk', oip := oip', osn := osn', sip := sip',
            handled := handled'|) }
    | _ ⇒ {}"

```

**lemma** *is\_rreq\_asm [dest!]*:

```

assumes "ξ' ∈ is_rreq ξ"
shows "(∃hops' rreqid' dip' dsn' dsk' oip' osn' sip' handled'.
      msg ξ = Rreq hops' rreqid' dip' dsn' dsk' oip' osn' sip' handled' ∧
      ξ' = ξ(|hops := hops', rreqid := rreqid', dip := dip', dsn := dsn',
            dsk := dsk', oip := oip', osn := osn', sip := sip',
            handled := handled'|))"
using assms unfolding is_rreq_def
by (cases "msg ξ") simp_all

```

**definition** *is\_rrep*

```

where "is_rrep ξ ≡ case msg ξ of
      Rrep hops' dip' dsn' oip' sip' ⇒
        { ξ(|hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip'|) }
    | _ ⇒ {}"

```

**lemma** *is\_rrep\_asm [dest!]*:

```

assumes "ξ' ∈ is_rrep ξ"
shows "(∃hops' dip' dsn' oip' sip'.
      msg ξ = Rrep hops' dip' dsn' oip' sip' ∧
      ξ' = ξ(|hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip'|))"
using assms unfolding is_rrep_def
by (cases "msg ξ") simp_all

```

**definition** *is\_rerr*

```

where "is_rerr ξ ≡ case msg ξ of
      Rerr dests' sip' ⇒ { ξ(|dests := dests', sip := sip'|) }
    | _ ⇒ {}"

```

**lemma** *is\_rerr\_asm [dest!]*:

```

assumes "ξ' ∈ is_rerr ξ"
shows "(∃dests' sip'.
      msg ξ = Rerr dests' sip' ∧
      ξ' = ξ(|dests := dests', sip := sip'|))"
using assms unfolding is_rerr_def
by (cases "msg ξ") simp_all

```

**lemmas** *is\_msg\_defs =*

is\_rerr\_def is\_rrep\_def is\_rreq\_def is\_pkt\_def is\_newpkt\_def

```
lemma is_msg_inv_ip [simp]:  
  "ξ' ∈ is_rerr ξ  ⇒ ip ξ' = ip ξ"  
  "ξ' ∈ is_rrep ξ  ⇒ ip ξ' = ip ξ"  
  "ξ' ∈ is_rreq ξ  ⇒ ip ξ' = ip ξ"  
  "ξ' ∈ is_pkt ξ   ⇒ ip ξ' = ip ξ"  
  "ξ' ∈ is_newpkt ξ ⇒ ip ξ' = ip ξ"  
unfolding is_msg_defs  
by (cases "msg ξ", clarsimp+)
```

```
lemma is_msg_inv_sn [simp]:  
  "ξ' ∈ is_rerr ξ  ⇒ sn ξ' = sn ξ"  
  "ξ' ∈ is_rrep ξ  ⇒ sn ξ' = sn ξ"  
  "ξ' ∈ is_rreq ξ  ⇒ sn ξ' = sn ξ"  
  "ξ' ∈ is_pkt ξ   ⇒ sn ξ' = sn ξ"  
  "ξ' ∈ is_newpkt ξ ⇒ sn ξ' = sn ξ"  
unfolding is_msg_defs  
by (cases "msg ξ", clarsimp+)
```

```
lemma is_msg_inv_rt [simp]:  
  "ξ' ∈ is_rerr ξ  ⇒ rt ξ' = rt ξ"  
  "ξ' ∈ is_rrep ξ  ⇒ rt ξ' = rt ξ"  
  "ξ' ∈ is_rreq ξ  ⇒ rt ξ' = rt ξ"  
  "ξ' ∈ is_pkt ξ   ⇒ rt ξ' = rt ξ"  
  "ξ' ∈ is_newpkt ξ ⇒ rt ξ' = rt ξ"  
unfolding is_msg_defs  
by (cases "msg ξ", clarsimp+)
```

```
lemma is_msg_inv_rreqs [simp]:  
  "ξ' ∈ is_rerr ξ  ⇒ rreqs ξ' = rreqs ξ"  
  "ξ' ∈ is_rrep ξ  ⇒ rreqs ξ' = rreqs ξ"  
  "ξ' ∈ is_rreq ξ  ⇒ rreqs ξ' = rreqs ξ"  
  "ξ' ∈ is_pkt ξ   ⇒ rreqs ξ' = rreqs ξ"  
  "ξ' ∈ is_newpkt ξ ⇒ rreqs ξ' = rreqs ξ"  
unfolding is_msg_defs  
by (cases "msg ξ", clarsimp+)
```

```
lemma is_msg_inv_store [simp]:  
  "ξ' ∈ is_rerr ξ  ⇒ store ξ' = store ξ"  
  "ξ' ∈ is_rrep ξ  ⇒ store ξ' = store ξ"  
  "ξ' ∈ is_rreq ξ  ⇒ store ξ' = store ξ"  
  "ξ' ∈ is_pkt ξ   ⇒ store ξ' = store ξ"  
  "ξ' ∈ is_newpkt ξ ⇒ store ξ' = store ξ"  
unfolding is_msg_defs  
by (cases "msg ξ", clarsimp+)
```

```
lemma is_msg_inv_sip [simp]:  
  "ξ' ∈ is_pkt ξ   ⇒ sip ξ' = sip ξ"  
  "ξ' ∈ is_newpkt ξ ⇒ sip ξ' = sip ξ"  
unfolding is_msg_defs  
by (cases "msg ξ", clarsimp+)
```

### 4.3.3 The protocol process

```
datatype pseq =  
  PAadv  
  | PNewPkt  
  | PPkt  
  | PRreq  
  | PRrep  
  | PRerr
```

```
fun nat_of_seqp :: "pseq ⇒ nat"
```

```

where
  "nat_of_seqp PAodv = 1"
| "nat_of_seqp PPkt = 2"
| "nat_of_seqp PNewPkt = 3"
| "nat_of_seqp PRreq = 4"
| "nat_of_seqp PRrep = 5"
| "nat_of_seqp PRerr = 6"

instantiation "pseqp" :: ord
begin
definition less_eq_seqp [iff]: "l1 ≤ l2 = (nat_of_seqp l1 ≤ nat_of_seqp l2)"
definition less_seqp [iff]: "l1 < l2 = (nat_of_seqp l1 < nat_of_seqp l2)"
instance ..
end

abbreviation AODV
where
  "AODV ≡ λ_. [[clear_locals]] call(PAodv)"

abbreviation PKT
where
  "PKT args ≡

  [[ξ. let (data, dip, oip) = args ξ in
    (clear_locals ξ) (| data := data, dip := dip, oip := oip |)]
  call(PPkt)"]

abbreviation NEWPKT
where
  "NEWPKT args ≡

  [[ξ. let (data, dip) = args ξ in
    (clear_locals ξ) (| data := data, dip := dip |)]
  call(PNewPkt)"]

abbreviation RREQ
where
  "RREQ args ≡

  [[ξ. let (hops, rreqid, dip, dsn, dsk, oip, osn, sip, handled) = args ξ in
    (clear_locals ξ) (| hops := hops, rreqid := rreqid, dip := dip,
      dsn := dsn, dsk := dsk, oip := oip,
      osn := osn, sip := sip, handled := handled |)]
  call(PRreq)"]

abbreviation RREP
where
  "RREP args ≡

  [[ξ. let (hops, dip, dsn, oip, sip) = args ξ in
    (clear_locals ξ) (| hops := hops, dip := dip, dsn := dsn,
      oip := oip, sip := sip |)]
  call(PRrep)"]

abbreviation RERR
where
  "RERR args ≡

  [[ξ. let (dests, sip) = args ξ in
    (clear_locals ξ) (| dests := dests, sip := sip |)]
  call(PRerr)"]

fun ΓAODV :: "(state, msg, pseqp, pseqp label) seqp_env"
where
  "ΓAODV PAodv = labelled PAodv (
  receive(λmsg' ξ. ξ (| msg := msg' |)).
  (
    ⟨is_newpkt⟩ NEWPKT(λξ. (data ξ, ip ξ))
  ⊕ ⟨is_pkt⟩ PKT(λξ. (data ξ, dip ξ, oip ξ))
  ⊕ ⟨is_rreq⟩

```

```

    [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, { }) )]]
    RREQ(λξ. (hops ξ, rreqid ξ, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, sip ξ, handled ξ))
⊕ ⟨is_rrep⟩
    [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, { }) )]]
    RREP(λξ. (hops ξ, dip ξ, dsn ξ, oip ξ, sip ξ))
⊕ ⟨is_rerr⟩
    [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ, { }) )]]
    RERR(λξ. (dests ξ, sip ξ))
)
⊕ ⟨λξ. { ξ (| dip := dip ) | dip. dip ∈ qD(store ξ) ∩ vD(rt ξ) }⟩
    [[ξ. ξ (| data := hd(σ_queue(store ξ, dip ξ)) )]]
    unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, ip ξ)).
    [[ξ. ξ (| store := the (drop (dip ξ) (store ξ)) )]]
    AODV()
    ▷ [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
        then Some (inc (sqn (rt ξ) rip)) else None) )]]
        [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
        [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
        [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
        [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
            then (dests ξ) rip else None) )]]
        groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)). AODV()
⊕ ⟨λξ. { ξ (| dip := dip )
    | dip. dip ∈ qD(store ξ) - vD(rt ξ) ∧ the (σ_p-flag(store ξ, dip)) = req }⟩
    [[ξ. ξ (| store := unsetRRF (store ξ) (dip ξ) )]]
    [[ξ. ξ (| sn := inc (sn ξ) )]]
    [[ξ. ξ (| rreqid := nrreqid (rreqs ξ) (ip ξ) )]]
    [[ξ. ξ (| rreqs := rreqs ξ ∪ { (ip ξ, rreqid ξ) } )]]
    broadcast(λξ. rreq(0, rreqid ξ, dip ξ, sqn (rt ξ) (dip ξ), sqnf (rt ξ) (dip ξ), ip ξ, sn ξ,
        ip ξ, False)). AODV())"

| "ΓAODV PNewPkt = labelled PNewPkt (
    ⟨ξ. dip ξ = ip ξ⟩
    deliver(λξ. data ξ).AODV()
⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
    [[ξ. ξ (| store := add (data ξ) (dip ξ) (store ξ) )]]
    AODV())"

| "ΓAODV PPkt = labelled PPkt (
    ⟨ξ. dip ξ = ip ξ⟩
    deliver(λξ. data ξ).AODV()
⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
    (
        ⟨ξ. dip ξ ∈ vD (rt ξ)⟩
        unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, oip ξ)).AODV()
        ▷
            [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
                then Some (inc (sqn (rt ξ) rip)) else None) )]]
            [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
            [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
            [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
            [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
                then (dests ξ) rip else None) )]]
            groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
⊕ ⟨ξ. dip ξ ∉ vD (rt ξ)⟩
    (
        ⟨ξ. dip ξ ∈ iD (rt ξ)⟩
        groupcast(λξ. the (precs (rt ξ) (dip ξ)), λξ. rerr([dip ξ ↦ sqn (rt ξ) (dip ξ)],
            ip ξ)).AODV()
⊕ ⟨ξ. dip ξ ∉ iD (rt ξ)⟩
        AODV()
    )
    )
))"

```

```

| "ΓAODV PRreq = labelled PRreq (
  ⟨ξ. (oip ξ, rreqid ξ) ∈ rreqs ξ⟩
  AODV()
  ⊕ ⟨ξ. (oip ξ, rreqid ξ) ∉ rreqs ξ⟩
  [[ξ. ξ (| rt := update (rt ξ) (oip ξ) (osn ξ, kno, val, hops ξ + 1, sip ξ, { }) )]]
  [[ξ. ξ (| rreqs := rreqs ξ ∪ {(oip ξ, rreqid ξ)} )]]
  (
    ⟨ξ. handled ξ = False⟩
    (
      ⟨ξ. dip ξ = ip ξ⟩
      [[ξ. ξ (| sn := max (sn ξ) (dsn ξ) )]]
      unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(0, dip ξ, sn ξ, oip ξ, ip ξ)).
        broadcast(λξ. rreq(hops ξ + 1, rreqid ξ, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, ip ξ, True)).
          AODV()
      ▷
      [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
        then Some (inc (sqn (rt ξ) rip)) else None) )]]
      [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
      [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
      [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
      [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
        then (dests ξ) rip else None) )]]
      groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
    ⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
    (
      ⟨ξ. dip ξ ∈ vD (rt ξ) ∧ dsn ξ ≤ sqn (rt ξ) (dip ξ) ∧ sqnf (rt ξ) (dip ξ) = kno⟩
      [[ξ. ξ (| rt := the (addpreRT (rt ξ) (dip ξ) {sip ξ}) )]]
      [[ξ. ξ (| rt := the (addpreRT (rt ξ) (oip ξ) {the (nhop (rt ξ) (dip ξ))}) )]]
      unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(the (dhops (rt ξ) (dip ξ)), dip ξ,
        sqn (rt ξ) (dip ξ), oip ξ, ip ξ)).
        broadcast(λξ. rreq(hops ξ + 1, rreqid ξ, dip ξ, dsn ξ,
          dsk ξ, oip ξ, osn ξ, ip ξ, True)).
          AODV()
      ▷
      [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
        then Some (inc (sqn (rt ξ) rip)) else None) )]]
      [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
      [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
      [[ξ. ξ (| pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
      [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ { })
        then (dests ξ) rip else None) )]]
      groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
    ⊕ ⟨ξ. dip ξ ∉ vD (rt ξ) ∨ sqn (rt ξ) (dip ξ) < dsn ξ ∨ sqnf (rt ξ) (dip ξ) = unk⟩
      broadcast(λξ. rreq(hops ξ + 1, rreqid ξ, dip ξ, max (sqn (rt ξ) (dip ξ)) (dsn ξ),
        dsk ξ, oip ξ, osn ξ, ip ξ, False)).
        AODV()
    )
  )
  ⊕ ⟨ξ. handled ξ = True⟩
  broadcast(λξ. rreq(hops ξ + 1, rreqid ξ, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, ip ξ, True)).
  AODV()
) )"

```

```

| "ΓAODV PRrep = labelled PRrep (
  ⟨ξ. rt ξ ≠ update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, { }) ⟩
  (
    [[ξ. ξ (| rt := update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, { }) )]]
    (
      ⟨ξ. oip ξ = ip ξ ⟩
      AODV()
    ⊕ ⟨ξ. oip ξ ≠ ip ξ ⟩
    (
      ⟨ξ. oip ξ ∈ vD (rt ξ) ⟩
      [[ξ. ξ (| rt := the (addpreRT (rt ξ) (dip ξ) {the (nhop (rt ξ) (oip ξ))}) )]]
    )
  )

```

```

[[ξ. ξ (| rt := the (addpreRT (rt ξ) (the (nhop (rt ξ) (dip ξ)))
                    {the (nhop (rt ξ) (oip ξ))}) ]]]
unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(hops ξ + 1, dip ξ, dsn ξ, oip ξ, ip ξ)).
AODV()
▷
[[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
                        then Some (inc (sqn (rt ξ) rip)) else None) ]]]
[[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
[[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
[[ξ. ξ (| pre := ∪{ the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
[[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ {})
                        then (dests ξ) rip else None) ]]]
groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()
⊕ ⟨ξ. oip ξ ∉ vD (rt ξ)⟩
AODV()
)
)
)
⊕ ⟨ξ. rt ξ = update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {})⟩
AODV()
)"

| "ΓAODV PRerr = labelled PRerr (
  [[ξ. ξ (| dests := (λrip. case (dests ξ) rip of None ⇒ None
                            | Some rsn ⇒ if rip ∈ vD (rt ξ) ∧ the (nhop (rt ξ) rip) = sip ξ
                                            ∧ sqn (rt ξ) rip < rsn then Some rsn else None) ]]]
  [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) )]]
  [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) )]]
  [[ξ. ξ (| pre := ∪{ the (precs (rt ξ) rip) | rip. rip ∈ dom (dests ξ) } )]]
  [[ξ. ξ (| dests := (λrip. if ((dests ξ) rip ≠ None ∧ the (precs (rt ξ) rip) ≠ {})
                            then (dests ξ) rip else None) ]]]
  groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)). AODV()"

declare ΓAODV.simps [simp del, code del]
lemmas ΓAODV.simps [simp, code] = ΓAODV.simps [simplified]

fun ΓAODV.skeleton
where
  "ΓAODV.skeleton PAadv = seqp_skeleton (ΓAODV PAadv)"
  | "ΓAODV.skeleton PNewPkt = seqp_skeleton (ΓAODV PNewPkt)"
  | "ΓAODV.skeleton PPkt = seqp_skeleton (ΓAODV PPkt)"
  | "ΓAODV.skeleton PRreq = seqp_skeleton (ΓAODV PRreq)"
  | "ΓAODV.skeleton PRrep = seqp_skeleton (ΓAODV PRrep)"
  | "ΓAODV.skeleton PRerr = seqp_skeleton (ΓAODV PRerr)"

lemma ΓAODV.skeleton_wf [simp]:
  "wellformed ΓAODV.skeleton"
proof (rule, intro allI)
  fix pn pn'
  show "call(pn') ∉ stermsl (ΓAODV.skeleton pn)"
  by (cases pn) simp_all
qed

declare ΓAODV.skeleton.simps [simp del, code del]
lemmas ΓAODV.skeleton.simps [simp, code]
  = ΓAODV.skeleton.simps [simplified ΓAODV.simps seqp_skeleton.simps]

lemma aodv_proc_cases [dest]:
  fixes p pn
  shows "p ∈ ctermsl (ΓAODV pn) ⇒
        (p ∈ ctermsl (ΓAODV PAadv) ∨
         p ∈ ctermsl (ΓAODV PNewPkt) ∨
         p ∈ ctermsl (ΓAODV PPkt) ∨
         p ∈ ctermsl (ΓAODV PRreq) ∨
         p ∈ ctermsl (ΓAODV PRrep) ∨
         p ∈ ctermsl (ΓAODV PRerr))"

```

```

      p ∈ cterms1 (ΓAODV PRrep) ∨
      p ∈ cterms1 (ΓAODV PRerr)"
using assms
by (cases pn) simp_all

definition σAODV :: "ip ⇒ (state × (state, msg, pseq, pseq label) seq) set"
where "σAODV i ≡ {(aodv_init i, ΓAODV PAodv)}"

abbreviation paodv
  :: "ip ⇒ (state × (state, msg, pseq, pseq label) seq, msg seq_action) automaton"
where
  "paodv i ≡ (| init = σAODV i, trans = seqp_sos ΓAODV |)"

lemma aodv_trans: "trans (paodv i) = seqp_sos ΓAODV"
  by simp

lemma aodv_control_within [simp]: "control_within ΓAODV (init (paodv i))"
  unfolding σAODV_def by (rule control_withinI) (auto simp del: ΓAODV_simps)

lemma aodv_wf [simp]:
  "wellformed ΓAODV"
proof (rule, intro allI)
  fix pn pn'
  show "call(pn') ∉ sterms1 (ΓAODV pn)"
  by (cases pn) simp_all
qed

lemmas aodv_labels_not_empty [simp] = labels_not_empty [OF aodv_wf]

lemma aodv_ex_label [intro]: "∃l. l ∈ labels ΓAODV p"
  by (metis aodv_labels_not_empty all_not_in_conv)

lemma aodv_ex_labelE [elim]:
  assumes "∀l ∈ labels ΓAODV p. P l p"
  and "∃p l. P l p ⇒ Q"
  shows "Q"
  using assms by (metis aodv_ex_label)

lemma aodv_simple_labels [simp]: "simple_labels ΓAODV"
proof
  fix pn p
  assume "p ∈ subterms(ΓAODV pn)"
  thus "∃!l. labels ΓAODV p = {l}"
  by (cases pn) (simp_all cong: seqp_congs | elim disjE)+
qed

lemma σAODV_labels [simp]: "(ξ, p) ∈ σAODV i ⇒ labels ΓAODV p = {PAodv-:0}"
  unfolding σAODV_def by simp

lemma aodv_init_kD_empty [simp]:
  "(ξ, p) ∈ σAODV i ⇒ kD (rt ξ) = {}"
  unfolding σAODV_def kD_def by simp

lemma aodv_init_sip_not_ip [simp]: "¬(sip (aodv_init i) = i)" by simp

lemma aodv_init_sip_not_ip' [simp]:
  assumes "(ξ, p) ∈ σAODV i"
  shows "sip ξ ≠ ip ξ"
  using assms unfolding σAODV_def by simp

lemma aodv_init_sip_not_i [simp]:
  assumes "(ξ, p) ∈ σAODV i"
  shows "sip ξ ≠ i"
  using assms unfolding σAODV_def by simp

```

```

lemma clear_locals_sip_not_ip':
  assumes "ip  $\xi = i$ "
  shows " $\neg(\text{sip } (\text{clear\_locals } \xi) = i)$ "
  using assms by auto

```

Stop the simplifier from descending into process terms.

```

declare seq_congs [cong]

```

Configure the main invariant tactic for AODV.

```

declare
   $\Gamma_{AODV\_simps}$  [cterms_env]
  aodv_proc_cases [ctermstl_cases]
  seq_invariant_ctermstI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans,
                           cterms_intros]
  seq_step_invariant_ctermstI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans,
                                cterms_intros]

```

```

end

```

## 4.4 Invariant assumptions and properties

```

theory D_Aodv_Predicates

```

```

imports D_Aodv

```

```

begin

```

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

```

abbreviation not_Pkt :: "msg  $\Rightarrow$  bool"

```

```

where "not_Pkt m  $\equiv$  case m of Pkt _ _ _  $\Rightarrow$  False | _  $\Rightarrow$  True"

```

```

definition msg_sender :: "msg  $\Rightarrow$  ip"

```

```

where "msg_sender m  $\equiv$  case m of Rreq _ _ _ _ _ ipc _  $\Rightarrow$  ipc
      | Rrep _ _ _ _ ipc  $\Rightarrow$  ipc
      | Rerr _ ipc  $\Rightarrow$  ipc
      | Pkt _ _ ipc  $\Rightarrow$  ipc"

```

```

lemma msg_sender_simps [simp]:

```

```

  " $\bigwedge$ hops rreqid dip dsn dsk oip osn sip handled.

```

```

    msg_sender (Rreq hops rreqid dip dsn dsk oip osn sip handled) = sip"

```

```

  " $\bigwedge$ hops dip dsn oip sip. msg_sender (Rrep hops dip dsn oip sip) = sip"

```

```

  " $\bigwedge$ dests sip. msg_sender (Rerr dests sip) = sip"

```

```

  " $\bigwedge$ dip sip. msg_sender (Pkt d dip sip) = sip"

```

```

  unfolding msg_sender_def by simp_all

```

```

definition msg_zhops :: "msg  $\Rightarrow$  bool"

```

```

where "msg_zhops m  $\equiv$  case m of

```

```

      Rreq hopsc _ dipc _ _ oipc _ sipc _  $\Rightarrow$  hopsc = 0  $\longrightarrow$  oipc = sipc

```

```

      | Rrep hopsc dipc _ _ sipc  $\Rightarrow$  hopsc = 0  $\longrightarrow$  dipc = sipc

```

```

      | _  $\Rightarrow$  True"

```

```

lemma msg_zhops_simps [simp]:

```

```

  " $\bigwedge$ hops rreqid dip dsn dsk oip osn sip handled.

```

```

    msg_zhops (Rreq hops rreqid dip dsn dsk oip osn sip handled) = (hops = 0  $\longrightarrow$  oip = sip)"

```

```

  " $\bigwedge$ hops dip dsn oip sip. msg_zhops (Rrep hops dip dsn oip sip) = (hops = 0  $\longrightarrow$  dip = sip)"

```

```

  " $\bigwedge$ dests sip. msg_zhops (Rerr dests sip) = True"

```

```

  " $\bigwedge$ dip. msg_zhops (Newpkt d dip) = True"

```

```

  " $\bigwedge$ dip sip. msg_zhops (Pkt d dip sip) = True"

```

```

  unfolding msg_zhops_def by simp_all

```

```

definition rreq_rrep_sn :: "msg  $\Rightarrow$  bool"

```

```

where "rreq_rrep_sn m  $\equiv$  case m of Rreq _ _ _ _ _ osnc _ _  $\Rightarrow$  osnc  $\geq$  1

```

```

      | Rrep _ _ dsnc _ _  $\Rightarrow$  dsnc  $\geq$  1

```

```

      | _  $\Rightarrow$  True"

```



lemma rreq\_rrep\_sn\_simps [simp]:

```
" $\wedge$ hops rreqid dip dsn dsk oip osn sip handled.
  rreq_rrep_sn (Rreq hops rreqid dip dsn dsk oip osn sip handled) = (osn  $\geq$  1)"
" $\wedge$ hops dip dsn oip sip. rreq_rrep_sn (Rrep hops dip dsn oip sip) = (dsn  $\geq$  1)"
" $\wedge$ dests sip. rreq_rrep_sn (Rerr dests sip) = True"
" $\wedge$ d dip. rreq_rrep_sn (Newpkt d dip) = True"
" $\wedge$ d dip sip. rreq_rrep_sn (Pkt d dip sip) = True"
unfolding rreq_rrep_sn_def by simp_all
```

definition rreq\_rrep\_fresh :: "rt  $\Rightarrow$  msg  $\Rightarrow$  bool"

```
where "rreq_rrep_fresh crt m  $\equiv$  case m of Rreq hopsc _ _ _ oipc osnc ipcc _  $\Rightarrow$  (ipcc  $\neq$  oipc  $\longrightarrow$ 
  oipc $\in$ kD(crt)  $\wedge$  (sqn crt oipc > osnc
     $\vee$  (sqn crt oipc = osnc
       $\wedge$  the (dhops crt oipc)  $\leq$  hopsc
       $\wedge$  the (flag crt oipc) = val)))
  | Rrep hopsc dipc dsnc _ ipcc  $\Rightarrow$  (ipcc  $\neq$  dipc  $\longrightarrow$ 
  dipc $\in$ kD(crt)
     $\wedge$  sqn crt dipc = dsnc
     $\wedge$  the (dhops crt dipc) = hopsc
     $\wedge$  the (flag crt dipc) = val)
  | _  $\Rightarrow$  True"
```

lemma rreq\_rrep\_fresh [simp]:

```
" $\wedge$ hops rreqid dip dsn dsk oip osn sip handled.
  rreq_rrep_fresh crt (Rreq hops rreqid dip dsn dsk oip osn sip handled) =
  (sip  $\neq$  oip  $\longrightarrow$  oip $\in$ kD(crt)
     $\wedge$  (sqn crt oip > osn
       $\vee$  (sqn crt oip = osn
         $\wedge$  the (dhops crt oip)  $\leq$  hops
         $\wedge$  the (flag crt oip) = val)))"
" $\wedge$ hops dip dsn oip sip. rreq_rrep_fresh crt (Rrep hops dip dsn oip sip) =
  (sip  $\neq$  dip  $\longrightarrow$  dip $\in$ kD(crt)
     $\wedge$  sqn crt dip = dsn
     $\wedge$  the (dhops crt dip) = hops
     $\wedge$  the (flag crt dip) = val)"
" $\wedge$ dests sip. rreq_rrep_fresh crt (Rerr dests sip) = True"
" $\wedge$ d dip. rreq_rrep_fresh crt (Newpkt d dip) = True"
" $\wedge$ d dip sip. rreq_rrep_fresh crt (Pkt d dip sip) = True"
unfolding rreq_rrep_fresh_def by simp_all
```

definition rerr\_invalid :: "rt  $\Rightarrow$  msg  $\Rightarrow$  bool"

```
where "rerr_invalid crt m  $\equiv$  case m of Rerr destsc _  $\Rightarrow$  ( $\forall$ ripc $\in$ dom(destsc).
  (ripc $\in$ iD(crt)  $\wedge$  the (destsc ripc) = sqn crt ripc))
  | _  $\Rightarrow$  True"
```

lemma rerr\_invalid [simp]:

```
" $\wedge$ hops rreqid dip dsn dsk oip osn sip handled.
  rerr_invalid crt (Rreq hops rreqid dip dsn dsk oip osn sip handled) = True"
" $\wedge$ hops dip dsn oip sip. rerr_invalid crt (Rrep hops dip dsn oip sip) = True"
" $\wedge$ dests sip. rerr_invalid crt (Rerr dests sip) = ( $\forall$ ripc $\in$ dom(dests).
  ripc $\in$ iD(crt)  $\wedge$  the (dests rip) = sqn crt rip)"
" $\wedge$ d dip. rerr_invalid crt (Newpkt d dip) = True"
" $\wedge$ d dip sip. rerr_invalid crt (Pkt d dip sip) = True"
unfolding rerr_invalid_def by simp_all
```

definition

```
initmissing :: "(nat  $\Rightarrow$  state option)  $\times$  'a  $\Rightarrow$  (nat  $\Rightarrow$  state)  $\times$  'a"
```

where

```
"initmissing  $\sigma$  = ( $\lambda$ i. case (fst  $\sigma$ ) i of None  $\Rightarrow$  aadv_init i | Some s  $\Rightarrow$  s, snd  $\sigma$ )"
```

lemma not\_in\_net\_ips\_fst\_init\_missing [simp]:

```
assumes "i  $\notin$  net_ips  $\sigma$ "
shows "fst (initmissing (netgmap fst  $\sigma$ )) i = aadv_init i"
```

```
using assms unfolding initmissing_def by simp
```

```
lemma fst_initmissing_netgmap_pair_fst [simp]:
  "fst (initmissing (netgmap ( $\lambda(p, q). (fst (id p), snd (id p), q)) s))
    = fst (initmissing (netgmap fst s))"
  unfolding initmissing_def by auto$ 
```

We introduce a streamlined alternative to `initmissing` with `netgmap` to simplify invariant statements and thus facilitate their comprehension and presentation.

```
lemma fst_initmissing_netgmap_default_aadv_init_netlift:
  "fst (initmissing (netgmap fst s)) = default aadv_init (netlift fst s)"
  unfolding initmissing_def default_def
  by (simp add: fst_netgmap_netlift del: One_nat_def)
```

**definition**

```
netglobal :: "(nat  $\Rightarrow$  state)  $\Rightarrow$  bool  $\Rightarrow$  ((state  $\times$  'b)  $\times$  'c) net_state  $\Rightarrow$  bool"
where
  "netglobal P  $\equiv$  ( $\lambda s. P (default aadv_init (netlift fst s))$ )"
```

end

## 4.5 Quality relations between routes

```
theory D_Fresher
imports D_Aadv_Data
begin
```

### 4.5.1 Net sequence numbers

On individual routes

**definition**

```
nsqn_r :: "r  $\Rightarrow$  sqn"
where
  "nsqn_r r  $\equiv$  if  $\pi_4(r) = \text{val} \vee \pi_2(r) = 0$  then  $\pi_2(r)$  else ( $\pi_2(r) - 1$ )"
```

**lemma nsqn\_r\_def'**:

```
"nsqn_r r = (if  $\pi_4(r) = \text{inv}$  then  $\pi_2(r) - 1$  else  $\pi_2(r)$ )"
  unfolding nsqn_r_def by simp
```

**lemma nsqn\_r\_zero [simp]**:

```
" $\bigwedge dsn dsk \text{flag hops nhip pre. nsqn}_r (0, dsk, \text{flag}, \text{hops}, \text{nhip}, \text{pre}) = 0$ "
  unfolding nsqn_r_def by clarsimp
```

**lemma nsqn\_r\_val [simp]**:

```
" $\bigwedge dsn dsk \text{hops nhip pre. nsqn}_r (dsn, dsk, \text{val}, \text{hops}, \text{nhip}, \text{pre}) = dsn$ "
  unfolding nsqn_r_def by clarsimp
```

**lemma nsqn\_r\_inv [simp]**:

```
" $\bigwedge dsn dsk \text{hops nhip pre. nsqn}_r (dsn, dsk, \text{inv}, \text{hops}, \text{nhip}, \text{pre}) = dsn - 1$ "
  unfolding nsqn_r_def by clarsimp
```

**lemma nsqn\_r\_lte\_dsn [simp]**:

```
" $\bigwedge dsn dsk \text{flag hops nhip pre. nsqn}_r (dsn, dsk, \text{flag}, \text{hops}, \text{nhip}, \text{pre}) \leq dsn$ "
  unfolding nsqn_r_def by clarsimp
```

On routes in routing tables

**definition**

```
nsqn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  sqn"
where
  "nsqn  $\equiv$   $\lambda rt dip. \text{case } \sigma_{\text{route}}(rt, dip) \text{ of None } \Rightarrow 0 \mid \text{Some } r \Rightarrow \text{nsqn}_r(r)$ "
```

**lemma nsqn\_sqn\_def**:

```

"∧rt dip. nsqn rt dip = (if flag rt dip = Some val ∨ sqn rt dip = 0
                        then sqn rt dip else sqn rt dip - 1)"
unfolding nsqn_def sqn_def by (clarsimp split: option.split) auto

lemma not_in_kD_nsqn [simp]:
  assumes "dip ∉ kD(rt)"
  shows "nsqn rt dip = 0"
  using assms unfolding nsqn_def by simp

lemma kD_nsqn:
  assumes "dip ∈ kD(rt)"
  shows "nsqn rt dip = nsqn_r(the (σ_route(rt, dip)))"
  using assms [THEN kD_Some] unfolding nsqn_def by clarsimp

lemma nsqn_r_flag_pred [simp, intro]:
  fixes dsn dsk flag hops nhip pre
  assumes "P (nsqn_r (dsn, dsk, val, hops, nhip, pre))"
  and "P (nsqn_r (dsn, dsk, inv, hops, nhip, pre))"
  shows "P (nsqn_r (dsn, dsk, flag, hops, nhip, pre))"
  using assms by (cases flag) auto

lemma nsqn_r_addpreRT_inv [simp]:
"∧rt dip npre dip'. dip ∈ kD(rt) ⇒
  nsqn_r (the (the (addpreRT rt dip npre) dip')) = nsqn_r (the (rt dip'))"
unfolding addpreRT_def nsqn_r_def
  by (frule kD_Some) (clarsimp split: option.split)

lemma sqn_nsqn:
"∧rt dip. sqn rt dip - 1 ≤ nsqn rt dip"
unfolding sqn_def nsqn_def by (clarsimp split: option.split)

lemma nsqn_sqn: "nsqn rt dip ≤ sqn rt dip"
unfolding sqn_def nsqn_def by (cases "rt dip") auto

lemma val_nsqn_sqn [elim, simp]:
  assumes "ip ∈ kD(rt)"
  and "the (flag rt ip) = val"
  shows "nsqn rt ip = sqn rt ip"
  using assms unfolding nsqn_sqn_def by auto

lemma vD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ vD(rt)"
  shows "nsqn rt ip = sqn rt ip"
  proof -
    from ⟨ip ∈ vD(rt)⟩ have "ip ∈ kD(rt)"
      and "the (flag rt ip) = val" by auto
    thus ?thesis ..
  qed

lemma inv_nsqn_sqn [elim, simp]:
  assumes "ip ∈ kD(rt)"
  and "the (flag rt ip) = inv"
  shows "nsqn rt ip = sqn rt ip - 1"
  using assms unfolding nsqn_sqn_def by auto

lemma iD_nsqn_sqn [elim, simp]:
  assumes "ip ∈ iD(rt)"
  shows "nsqn rt ip = sqn rt ip - 1"
  proof -
    from ⟨ip ∈ iD(rt)⟩ have "ip ∈ kD(rt)"
      and "the (flag rt ip) = inv" by auto
    thus ?thesis ..
  qed

```

```

lemma nsqn_update_changed_kno_val [simp]: "\rt ip dsn dsk hops nhip.
  rt  $\neq$  update rt ip (dsn, kno, val, hops, nhip, { })
   $\implies$  nsqn (update rt ip (dsn, kno, val, hops, nhip, { })) ip = dsn"
using assms unfolding nsqn_r_def update_def
by (clarsimp simp: kD_nsqn split: option.split_asm option.split split_if_asm)
  (metis fun_upd_triv)

```

```

lemma nsqn_addpreRT_inv [simp]:
  "\rt dip npre dip'. dip  $\in$  kD(rt)  $\implies$ 
  nsqn (the (addpreRT rt dip npre)) dip' = nsqn rt dip'"
unfolding addpreRT_def nsqn_def nsqn_r_def
by (frule kD_Some) (clarsimp split: option.split)

```

```

lemma nsqn_update_other [simp]:
  fixes dsn dsk flag hops dip nhip pre rt ip
  assumes "dip  $\neq$  ip"
  shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip, pre)) dip = nsqn rt dip"
using assms unfolding nsqn_def
by (clarsimp split: option.split)

```

```

lemma nsqn_invalidate_eq:
  assumes "dip  $\in$  kD(rt)"
  and "dests dip = Some rsn"
  shows "nsqn (invalidate rt dests) dip = rsn - 1"
using assms
proof -
  from assms obtain dsk hops nhip pre
  where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip, pre)"
  unfolding invalidate_def
  by auto
  moreover from (dip  $\in$  kD(rt)) have "dip  $\in$  kD(invalidate rt dests)" by simp
  ultimately show ?thesis
  using (dests dip = Some rsn) by simp
qed

```

```

lemma nsqn_invalidate_other [simp]:
  assumes "dip  $\in$  kD(rt)"
  and "dip  $\notin$  dom dests"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
using assms by (clarsimp simp add: kD_nsqn)

```

## 4.5.2 Comparing routes

definition

```

fresher :: "r  $\Rightarrow$  r  $\Rightarrow$  bool" ("(_/  $\sqsubseteq$  _)" [51, 51] 50)

```

where

```

"fresher r r'  $\equiv$  ((nsqn_r r < nsqn_r r')  $\vee$  (nsqn_r r = nsqn_r r'  $\wedge$   $\pi_5(r) \geq \pi_5(r')$ ))"

```

```

lemma fresherI1 [intro]:
  assumes "nsqn_r r < nsqn_r r'"
  shows "r  $\sqsubseteq$  r'"
unfolding fresher_def using assms by simp

```

```

lemma fresherI2 [intro]:
  assumes "nsqn_r r = nsqn_r r'"
  and " $\pi_5(r) \geq \pi_5(r')$ "
  shows "r  $\sqsubseteq$  r'"
unfolding fresher_def using assms by simp

```

```

lemma fresherI [intro]:
  assumes "(nsqn_r r < nsqn_r r')  $\vee$  (nsqn_r r = nsqn_r r'  $\wedge$   $\pi_5(r) \geq \pi_5(r')$ )"
  shows "r  $\sqsubseteq$  r'"
unfolding fresher_def using assms .

```

```

lemma fresherE [elim]:
  assumes "r  $\sqsubseteq$  r'"
    and "nsqnr r < nsqnr r'  $\implies$  P r r'"
    and "nsqnr r = nsqnr r'  $\wedge$   $\pi_5(r) \geq \pi_5(r') \implies$  P r r'"
  shows "P r r'"
using assms unfolding fresher_def by auto

```

```

lemma fresher_refl [simp]: "r  $\sqsubseteq$  r"
unfolding fresher_def by simp

```

```

lemma fresher_trans [elim, trans]:
  "[[ x  $\sqsubseteq$  y; y  $\sqsubseteq$  z ]  $\implies$  x  $\sqsubseteq$  z"
unfolding fresher_def by auto

```

```

lemma not_fresher_trans [elim, trans]:
  "[[  $\neg(x \sqsubseteq y)$ ;  $\neg(z \sqsubseteq x)$  ]  $\implies$   $\neg(z \sqsubseteq y)$ "
unfolding fresher_def by auto

```

```

lemma fresher_dsn_flag_hops_const [simp]:
  fixes dsn dsk dsk' flag hops nhip nhip' pre pre'
  shows "(dsn, dsk, flag, hops, nhip, pre)  $\sqsubseteq$  (dsn, dsk', flag, hops, nhip', pre)"
unfolding fresher_def by (cases flag) simp_all

```

```

lemma addpre_fresher [simp]: " $\bigwedge r$  npre. r  $\sqsubseteq$  (addpre r npre)"
by clarsimp

```

### 4.5.3 Comparing routing tables

definition

```
rt_fresher :: "ip  $\Rightarrow$  rt  $\Rightarrow$  rt  $\Rightarrow$  bool"
```

where

```
"rt_fresher  $\equiv$   $\lambda$ dip rt rt'. (the ( $\sigma_{route}(rt, dip)$ ))  $\sqsubseteq$  (the ( $\sigma_{route}(rt', dip)$ ))"
```

abbreviation

```
rt_fresher_syn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  rt  $\Rightarrow$  bool" ("(_/  $\sqsubseteq$  _)" [51, 999, 51] 50)
```

where

```
"rt1  $\sqsubseteq_i$  rt2  $\equiv$  rt_fresher i rt1 rt2"
```

lemma rt\_fresher\_def':

```
"(rt1  $\sqsubseteq_i$  rt2) = (nsqnr (the (rt1 i)) < nsqnr (the (rt2 i))  $\vee$ 
  nsqnr (the (rt1 i)) = nsqnr (the (rt2 i))  $\wedge$   $\pi_5$  (the (rt2 i))  $\leq$   $\pi_5$  (the (rt1 i)))"
```

```
unfolding rt_fresher_def fresher_def by (rule refl)
```

lemma single\_rt\_fresher [intro]:

```
assumes "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"
```

```
shows "rt1  $\sqsubseteq_{ip}$  rt2"
```

```
using assms unfolding rt_fresher_def .
```

lemma rt\_fresher\_single [intro]:

```
assumes "rt1  $\sqsubseteq_{ip}$  rt2"
```

```
shows "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"
```

```
using assms unfolding rt_fresher_def .
```

lemma rt\_fresher\_def2:

```
assumes "dip  $\in$  kD(rt1)"
```

```
and "dip  $\in$  kD(rt2)"
```

```
shows "(rt1  $\sqsubseteq_{dip}$  rt2) = (nsqn rt1 dip < nsqn rt2 dip
   $\vee$  (nsqn rt1 dip = nsqn rt2 dip
     $\wedge$  the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip)))"
```

```
using assms unfolding rt_fresher_def fresher_def by (simp add: kD_nsqn proj5_eq_dhops)
```

lemma rt\_fresherI1 [intro]:

```
assumes "dip  $\in$  kD(rt1)"
```

```
and "dip  $\in$  kD(rt2)"
```

```

    and "nsqn rt1 dip < nsqn rt2 dip"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
  unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt_fresherI2 [intro]:
  assumes "dip  $\in$  kD(rt1)"
    and "dip  $\in$  kD(rt2)"
    and "nsqn rt1 dip = nsqn rt2 dip"
    and "the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip)"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
  unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3-4) by simp

lemma rt_fresherE [elim]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "dip  $\in$  kD(rt1)"
    and "dip  $\in$  kD(rt2)"
    and "[[ nsqn rt1 dip < nsqn rt2 dip ]  $\implies$  P rt1 rt2 dip]"
    and "[[ nsqn rt1 dip = nsqn rt2 dip;
      the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip) ]  $\implies$  P rt1 rt2 dip]"
  shows "P rt1 rt2 dip"
  using assms(1) unfolding rt_fresher_def2 [OF assms(2-3)]
  using assms(4-5) by auto

lemma rt_fresher_refl [simp]: "rt  $\sqsubseteq_{dip}$  rt"
  unfolding rt_fresher_def by simp

lemma rt_fresher_trans [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
  using assms unfolding rt_fresher_def by auto

lemma rt_fresher_if_Some [intro!]:
  assumes "the (rt dip)  $\sqsubseteq$  r"
  shows "rt  $\sqsubseteq_{dip}$  ( $\lambda$ ip. if ip = dip then Some r else rt ip)"
  using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip  $\Rightarrow$  rt  $\Rightarrow$  rt  $\Rightarrow$  bool"
where
  "rt_fresh_as  $\equiv$   $\lambda$ dip rt1 rt2. (rt1  $\sqsubseteq_{dip}$  rt2)  $\wedge$  (rt2  $\sqsubseteq_{dip}$  rt1)"

abbreviation
  rt_fresh_as_syn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  rt  $\Rightarrow$  bool" ("(_/  $\approx$  _)" [51, 999, 51] 50)
where
  "rt1  $\approx_i$  rt2  $\equiv$  rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_refl [simp]: " $\wedge$ rt dip. rt  $\approx_{dip}$  rt"
  unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
  " $\wedge$ rt1 rt2 rt3 dip. [[ rt1  $\approx_{dip}$  rt2; rt2  $\approx_{dip}$  rt3 ]  $\implies$  rt1  $\approx_{dip}$  rt3"
  unfolding rt_fresh_as_def rt_fresher_def
  by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt1"
  shows "rt1  $\approx_{dip}$  rt2"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
  assumes "dip  $\in$  kD(rt1)"
    and "dip  $\in$  kD(rt2)"
    and "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"

```

```

    and "the (rt2 dip)  $\sqsubseteq$  the (rt1 dip)"
    shows "rt1  $\approx_{dip}$  rt2"
using assms unfolding rt_fresh_as_def
by (clarsimp dest!: single_rt_fresher)

lemma nsqn_rt_fresh_asI:
  assumes "dip  $\in$  kD(rt)"
    and "dip  $\in$  kD(rt')"
    and "nsqn rt dip = nsqn rt' dip"
    and " $\pi_5(\text{the } (rt \text{ dip})) = \pi_5(\text{the } (rt' \text{ dip}))$ "
  shows "rt  $\approx_{dip}$  rt'"
proof
  from assms(1-2,4) have dhops': "the (dhops rt' dip)  $\leq$  the (dhops rt dip)"
  by (simp add: proj5_eq_dhops)
  with assms(1-3) show "rt  $\sqsubseteq_{dip}$  rt'"
  by (rule rt_fresherI2)
next
  from assms(1-2,4) have dhops: "the (dhops rt dip)  $\leq$  the (dhops rt' dip)"
  by (simp add: proj5_eq_dhops)
  with assms(2,1) assms(3) [symmetric] show "rt'  $\sqsubseteq_{dip}$  rt"
  by (rule rt_fresherI2)
qed

lemma rt_fresh_asE [elim]:
  assumes "rt1  $\approx_{dip}$  rt2"
    and "[[ rt1  $\sqsubseteq_{dip}$  rt2; rt2  $\sqsubseteq_{dip}$  rt1 ]  $\implies$  P rt1 rt2 dip]"
  shows "P rt1 rt2 dip"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD1 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\sqsubseteq_{dip}$  rt1"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\approx_{dip}$  rt1"
using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
  assumes " $\neg$  (rt1  $\sqsubseteq_{dip}$  rt2)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
proof
  assume "rt1  $\approx_{dip}$  rt2"
  hence "rt1  $\sqsubseteq_{dip}$  rt2" ..
  with ( $\neg$  (rt1  $\sqsubseteq_{dip}$  rt2)) show False ..
qed

lemma not_rt_fresh_asI2 [intro]:
  assumes " $\neg$  (rt2  $\sqsubseteq_{dip}$  rt1)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
proof
  assume "rt1  $\approx_{dip}$  rt2"
  hence "rt2  $\sqsubseteq_{dip}$  rt1" ..
  with ( $\neg$  (rt2  $\sqsubseteq_{dip}$  rt1)) show False ..
qed

lemma not_single_rt_fresher [elim]:
  assumes " $\neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))"

```

```

  shows "¬(rt1  $\sqsubseteq_{ip}$  rt2)"
proof
  assume "rt1  $\sqsubseteq_{ip}$  rt2"
  hence "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)" ..
  with (¬(the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))) show False ..
qed

lemmas not_single_rt_fresh_asI1 [intro] = not_rt_fresh_asI1 [OF not_single_rt_fresher]
lemmas not_single_rt_fresh_asI2 [intro] = not_rt_fresh_asI2 [OF not_single_rt_fresher]

lemma not_rt_fresher_single [elim]:
  assumes "¬(rt1  $\sqsubseteq_{ip}$  rt2)"
  shows "¬(the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))"
proof
  assume "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"
  hence "rt1  $\sqsubseteq_{ip}$  rt2" ..
  with (¬(rt1  $\sqsubseteq_{ip}$  rt2)) show False ..
qed

lemma rt_fresh_as_nsqnr:
  assumes "dip  $\in$  kD(rt1)"
  and "dip  $\in$  kD(rt2)"
  and "rt1  $\approx_{dip}$  rt2"
  shows "nsqnr (the (rt2 dip)) = nsqnr (the (rt1 dip))"
using assms(3) unfolding rt_fresh_as_def
by (auto simp: rt_fresher_def2 [OF ⟨dip  $\in$  kD(rt1)⟩ ⟨dip  $\in$  kD(rt2)⟩]
    rt_fresher_def2 [OF ⟨dip  $\in$  kD(rt2)⟩ ⟨dip  $\in$  kD(rt1)⟩]
    kD_nsqn [OF ⟨dip  $\in$  kD(rt1)⟩]
    kD_nsqn [OF ⟨dip  $\in$  kD(rt2)⟩])

lemma rt_fresher_mapupd [intro!]:
  assumes "dip $\in$ kD(rt)"
  and "the (rt dip)  $\sqsubseteq$  r"
  shows "rt  $\sqsubseteq_{dip}$  rt(dip  $\mapsto$  r)"
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_map_update_other [intro!]:
  assumes "dip $\in$ kD(rt)"
  and "dip  $\neq$  ip"
  shows "rt  $\sqsubseteq_{dip}$  rt(ip  $\mapsto$  r)"
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_update_other [simp]:
  assumes inkD: "dip $\in$ kD(rt)"
  and "dip  $\neq$  ip"
  shows "rt  $\sqsubseteq_{dip}$  update rt ip r"
using assms unfolding update_def
by (clarsimp split: option.split) (fastforce)

theorem rt_fresher_update [simp]:
  assumes "dip $\in$ kD(rt)"
  and "the (dhops rt dip)  $\geq$  1"
  and "update_arg_wf r"
  shows "rt  $\sqsubseteq_{dip}$  update rt ip r"
proof (cases "dip = ip")
  assume "dip  $\neq$  ip" with ⟨dip $\in$ kD(rt)⟩ show ?thesis
  by (rule rt_fresher_update_other)
next
  assume "dip = ip"

  from ⟨dip $\in$ kD(rt)⟩ obtain dsnn dskn fn hopsn nhipn pren
  where rtn [simp]: "the (rt dip) = (dsnn, dskn, fn, hopsn, nhipn, pren)"
  by (metis prod_cases6)
  with ⟨the (dhops rt dip)  $\geq$  1⟩ and ⟨dip $\in$ kD(rt)⟩ have "hopsn  $\geq$  1"

```



```

by (metis proj5_eq_dhops projs(4))
from ⟨dip∈kD(rt)⟩ rtn have [simp]: "sqn rt dip = dsnn"
      and [simp]: "the (dhops rt dip) = hopsn"
      and [simp]: "the (flag rt dip) = fn"
by (simp add: sqn_def proj5_eq_dhops [symmetric]
      proj4_eq_flag [symmetric])
from ⟨update_arg_wf r⟩ have "(dsnn, dskn, fn, hopsn, nhipn, pren)
      ⊆ the ((update rt dip r) dip)"
proof (rule wf_r_cases)
  fix nhip pre
  from ⟨hopsn ≥ 1⟩ have "∧pre'. (dsnn, dskn, fn, hopsn, nhipn, pren)
      ⊆ (dsnn, unk, val, Suc 0, nhip, pre')"
  unfolding fresher_def sqn_def by (cases fn) auto
  thus "(dsnn, dskn, fn, hopsn, nhipn, pren)
      ⊆ the (update rt dip (0, unk, val, Suc 0, nhip, pre) dip)"
  using ⟨dip∈kD(rt)⟩ by - (rule update_cases_kD, simp_all)
next
fix dsn :: sqn and hops nhip pre
assume "0 < dsn"
show "(dsnn, dskn, fn, hopsn, nhipn, pren)
      ⊆ the (update rt dip (dsn, kno, val, hops, nhip, pre) dip)"
proof (rule update_cases_kD [OF _ ⟨dip∈kD(rt)⟩], simp_all add: ⟨0 < dsn⟩)
  assume "dsnn < dsn"
  thus "(dsnn, dskn, fn, hopsn, nhipn, pren)
      ⊆ (dsn, kno, val, hops, nhip, pre ∪ pren)"
  unfolding fresher_def by auto
next
assume "dsnn = dsn"
and "hops < hopsn"
thus "(dsn, dskn, fn, hopsn, nhipn, pren)
      ⊆ (dsn, kno, val, hops, nhip, pre ∪ pren)"
unfolding fresher_def nsqn_r_def by simp
next
assume "dsnn = dsn"
with ⟨0 < dsn⟩
show "(dsn, dskn, inv, hopsn, nhipn, pren)
      ⊆ (dsn, kno, val, hops, nhip, pre ∪ pren)"
unfolding fresher_def by simp
qed
qed
hence "rt ⊆dip update rt dip r"
by - (rule single_rt_fresher, simp)
with ⟨dip = ip⟩ show ?thesis by simp
qed

```

```

theorem rt_fresher_invalidate [simp]:
  assumes "dip∈kD(rt)"
  and indests: "∀rip∈dom(dests). rip∈vD(rt) ∧ sqn rt rip < the (dests rip)"
  shows "rt ⊆dip invalidate rt dests"
proof (cases "dip∈dom(dests)")
  assume "dip∉dom(dests)"
  thus ?thesis using ⟨dip∈kD(rt)⟩
  by - (rule single_rt_fresher, simp)
next
assume "dip∈dom(dests)"
moreover with indests have "dip∈vD(rt)"
      and "sqn rt dip < the (dests dip)"
  by auto
ultimately show ?thesis
  unfolding invalidate_def sqn_def
  by - (rule single_rt_fresher, auto simp: fresher_def)
qed

```

```

lemma nsqnr_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
    and "dip ∈ dom(dests)"
  shows "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
  using assms unfolding invalidate_def by auto

lemma rt_fresh_as_inc_invalidate [simp]:
  assumes "dip ∈ kD(rt)"
    and "∀ rip ∈ dom(dests). rip ∈ vD(rt) ∧ the (dests rip) = inc (sqn rt rip)"
  shows "rt ≈dip invalidate rt dests"
proof (cases "dip ∈ dom(dests)")
  assume "dip ∉ dom(dests)"
  with ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)"
    by simp
  with ⟨dip ∈ kD(rt)⟩ show ?thesis
    by rule (simp_all add: ⟨dip ∉ dom(dests)⟩)
next
  assume "dip ∈ dom(dests)"
  with assms(2) have "dip ∈ vD(rt)"
    and "the (dests dip) = inc (sqn rt dip)" by auto
  from ⟨dip ∈ vD(rt)⟩ have "dip ∈ kD(rt)" by simp
  moreover then have "dip ∈ kD(invalidate rt dests)" by simp
  ultimately show ?thesis
proof (rule nsqnr_fresh_asI)
  from ⟨dip ∈ vD(rt)⟩ have "nsqn rt dip = sqn rt dip" by simp
  also have "sqn rt dip = nsqnr (the (invalidate rt dests dip))"
  proof -
    from ⟨dip ∈ kD(rt)⟩ have "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
      using ⟨dip ∈ dom(dests)⟩ by (rule nsqnr_invalidate)
    with ⟨the (dests dip) = inc (sqn rt dip)⟩
      show "sqn rt dip = nsqnr (the (invalidate rt dests dip))" by simp
  qed
  also from ⟨dip ∈ kD(invalidate rt dests)⟩
    have "nsqnr (the (invalidate rt dests dip)) = nsqn (invalidate rt dests) dip"
      by (simp add: kD_nsqn)
  finally show "nsqn rt dip = nsqn (invalidate rt dests) dip" .
qed simp
qed

```

```

lemmas rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]

```

```

lemma rt_fresh_as_addpreRT [simp]:
  assumes "ip ∈ kD(rt)"
  shows "rt ≈dip the (addpreRT rt ip npre)"
  using assms [THEN kD_Some] by (auto simp: addpreRT_def)

```

```

lemmas rt_fresher_addpreRT [simp] = rt_fresh_as_addpreRT [THEN rt_fresh_asD1]

```

#### 4.5.4 Strictly comparing routing tables

```

definition rt_strictly_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"

```

where

```

"rt_strictly_fresher ≡ λdip rt1 rt2. (rt1 ⊆dip rt2) ∧ ¬(rt1 ≈dip rt2)"

```

abbreviation

```

rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊆i _)" [51, 999, 51] 50)

```

where

```

"rt1 ⊆i rt2 ≡ rt_strictly_fresher i rt1 rt2"

```

```

lemma rt_strictly_fresher_def'':

```

```

"rt1 ⊆i rt2 = ((rt1 ⊆i rt2) ∧ ¬(rt2 ⊆i rt1))"

```

```

unfolding rt_strictly_fresher_def rt_fresh_as_def by auto

```

```

lemma rt_strictly_fresherI' [intro]:

```

```

assumes "rt1  $\sqsubseteq_i$  rt2"
  and " $\neg$ (rt2  $\sqsubseteq_i$  rt1)"
  shows "rt1  $\sqsubseteq_i$  rt2"
using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherE' [elim]:
  assumes "rt1  $\sqsubseteq_i$  rt2"
  and "[[ rt1  $\sqsubseteq_i$  rt2;  $\neg$ (rt2  $\sqsubseteq_i$  rt1) ]  $\implies$  P rt1 rt2 i"
  shows "P rt1 rt2 i"
using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherI [intro]:
  assumes "rt1  $\sqsubseteq_i$  rt2"
  and " $\neg$ (rt1  $\approx_i$  rt2)"
  shows "rt1  $\sqsubseteq_i$  rt2"
unfolding rt_strictly_fresher_def using assms ..

lemmas rt_strictly_fresher_singleI [elim] = rt_strictly_fresherI [OF single_rt_fresher]

lemma rt_strictly_fresherE [elim]:
  assumes "rt1  $\sqsubseteq_i$  rt2"
  and "[[ rt1  $\sqsubseteq_i$  rt2;  $\neg$ (rt1  $\approx_i$  rt2) ]  $\implies$  P rt1 rt2 i"
  shows "P rt1 rt2 i"
using assms(1) unfolding rt_strictly_fresher_def
by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':
  "rt1  $\sqsubseteq_i$  rt2 =
  (nsqnr (the (rt1 i)) < nsqnr (the (rt2 i))
   $\vee$  (nsqnr (the (rt1 i)) = nsqnr (the (rt2 i))  $\wedge$   $\pi_5$ (the (rt1 i)) >  $\pi_5$ (the (rt2 i))))"
unfolding rt_strictly_fresher_def'' rt_fresher_def fresher_def by auto

lemma rt_strictly_fresher_fresherD [dest]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  shows "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"
using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  shows " $\neg$  rt1  $\approx_{dip}$  rt2"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
using assms proof -
  from (rt1  $\sqsubseteq_{dip}$  rt2) obtain "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)" by auto
  also from (rt2  $\sqsubseteq_{dip}$  rt3) obtain "the (rt2 dip)  $\sqsubseteq$  the (rt3 dip)" by auto
  finally have "the (rt1 dip)  $\sqsubseteq$  the (rt3 dip)" .

  moreover have " $\neg$  (rt1  $\approx_{dip}$  rt3)"
  proof -
    from (rt1  $\sqsubseteq_{dip}$  rt2) obtain " $\neg$ (the (rt2 dip)  $\sqsubseteq$  the (rt1 dip))" by auto
    also from (rt2  $\sqsubseteq_{dip}$  rt3) obtain " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt2 dip))" by auto
    finally have " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt1 dip))" .
    thus ?thesis ..
  qed
  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: " $\neg$  (rt  $\sqsubseteq_{dip}$  rt)"
using assms unfolding rt_strictly_fresher_def
by clarsimp

```

```

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
proof -
  from  $\langle rt1 \sqsubseteq_{dip} rt2 \rangle$  have "rt1  $\sqsubseteq_{dip}$  rt2"
    and " $\neg(rt2 \sqsubseteq_{dip} rt1)$ "
  unfolding rt_strictly_fresher_def'' by auto
  from this(1) and  $\langle rt2 \sqsubseteq_{dip} rt3 \rangle$  have "rt1  $\sqsubseteq_{dip}$  rt3" ..

  moreover from  $\langle \neg(rt2 \sqsubseteq_{dip} rt1) \rangle$  have " $\neg(rt3 \sqsubseteq_{dip} rt1)$ "
  proof (rule contrapos_nn)
    assume "rt3  $\sqsubseteq_{dip}$  rt1"
    with  $\langle rt2 \sqsubseteq_{dip} rt3 \rangle$  show "rt2  $\sqsubseteq_{dip}$  rt1" ..
  qed

  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3"
  unfolding rt_strictly_fresher_def'' by auto
qed

```

```

lemma rt_fresher_trans_rt_strictly_fresher' [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
proof -
  from  $\langle rt2 \sqsubseteq_{dip} rt3 \rangle$  have "rt2  $\sqsubseteq_{dip}$  rt3"
    and " $\neg(rt3 \sqsubseteq_{dip} rt2)$ "
  unfolding rt_strictly_fresher_def'' by auto
  from  $\langle rt1 \sqsubseteq_{dip} rt2 \rangle$  and this(1) have "rt1  $\sqsubseteq_{dip}$  rt3" ..

  moreover from  $\langle \neg(rt3 \sqsubseteq_{dip} rt2) \rangle$  have " $\neg(rt3 \sqsubseteq_{dip} rt1)$ "
  proof (rule contrapos_nn)
    assume "rt3  $\sqsubseteq_{dip}$  rt1"
    thus "rt3  $\sqsubseteq_{dip}$  rt2" using  $\langle rt1 \sqsubseteq_{dip} rt2 \rangle$  ..
  qed

  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3"
  unfolding rt_strictly_fresher_def'' by auto
qed

```

```

lemma rt_fresher_imp_nsqn_le:
  assumes "rt1  $\sqsubseteq_{ip}$  rt2"
    and "ip  $\in kD$  rt1"
    and "ip  $\in kD$  rt2"
  shows "nsqn rt1 ip  $\leq$  nsqn rt2 ip"
using assms(1)
by (auto simp add: rt_fresher_def2 [OF assms(2-3)])

```

```

lemma rt_strictly_fresher_ltI [intro]:
  assumes "dip  $\in kD(rt1)$ "
    and "dip  $\in kD(rt2)$ "
    and "nsqn rt1 dip < nsqn rt2 dip"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
proof
  from assms show "rt1  $\sqsubseteq_{dip}$  rt2" ..
next
  show " $\neg(rt1 \approx_{dip} rt2)$ "
  proof
    assume "rt1  $\approx_{dip}$  rt2"
    hence "rt2  $\sqsubseteq_{dip}$  rt1" ..
    hence "nsqn rt2 dip  $\leq$  nsqn rt1 dip"
      using  $\langle dip \in kD(rt2) \rangle$   $\langle dip \in kD(rt1) \rangle$ 
      by (rule rt_fresher_imp_nsqn_le)
  qed

```

```

    with ⟨nsqn rt1 dip < nsqn rt2 dip⟩ show "False"
      by simp
qed
qed

lemma rt_strictly_fresher_eqI [intro]:
  assumes "i ∈ kD(rt1)"
    and "i ∈ kD(rt2)"
    and "nsqn rt1 i = nsqn rt2 i"
    and "π5(the (rt2 i)) < π5(the (rt1 i))"
  shows "rt1 ⊑i rt2"
using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

lemma invalidate_rtsf_left [simp]:
  "∧dests dip rt rt'. dests dip = None ⇒ (invalidate rt dests ⊑dip rt') = (rt ⊑dip rt')"
unfolding invalidate_def rt_strictly_fresher_def'
by (rule iffI) (auto split: option.split_asm)

lemma vD_invalidate_rt_strictly_fresher [simp]:
  assumes "dip ∈ vD(invalidate rt1 dests)"
  shows "(invalidate rt1 dests ⊑dip rt2) = (rt1 ⊑dip rt2)"
proof (cases "dip ∈ dom(dests)")
  assume "dip ∈ dom(dests)"
  hence "dip ∉ vD(invalidate rt1 dests)"
    unfolding invalidate_def vD_def
    by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
  with ⟨dip ∈ vD(invalidate rt1 dests)⟩ show ?thesis by simp
next
  assume "dip ∉ dom(dests)"
  hence "dests dip = None" by auto
  moreover with ⟨dip ∈ vD(invalidate rt1 dests)⟩ have "dip ∈ vD(rt1)"
    unfolding invalidate_def vD_def
    by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
  ultimately show ?thesis
    unfolding invalidate_def rt_strictly_fresher_def' by auto
qed

lemma rt_strictly_fresher_update_other [elim!]:
  "∧dip ip rt r rt'. [ dip ≠ ip; rt ⊑dip rt' ] ⇒ update rt ip r ⊑dip rt'"
unfolding rt_strictly_fresher_def' by clarsimp

lemma addpreRT_strictly_fresher [simp]:
  assumes "dip ∈ kD(rt)"
  shows "(the (addpreRT rt dip npre) ⊑ip rt2) = (rt ⊑ip rt2)"
using assms unfolding rt_strictly_fresher_def' by clarsimp

lemma lt_sqn_imp_update_strictly_fresher:
  assumes "dip ∈ vD (rt2 nhip)"
    and *: "osn < sqn (rt2 nhip) dip"
    and **: "rt ≠ update rt dip (osn, kno, val, hops, nhip, {})"
  shows "update rt dip (osn, kno, val, hops, nhip, {}) ⊑dip rt2 nhip"
unfolding rt_strictly_fresher_def'
proof (rule disjI1)
  from ** have "nsqn (update rt dip (osn, kno, val, hops, nhip, {})) dip = osn"
    by (rule nsqn_update_changed_kno_val)
  with ⟨dip ∈ vD(rt2 nhip)⟩
  have "nsqnr (the (update rt dip (osn, kno, val, hops, nhip, {}) dip)) = osn"
    by (simp add: kD_nsqn)
  also have "osn < sqn (rt2 nhip) dip" by (rule *)
  also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip dip))"
    unfolding nsqnr_def using ⟨dip ∈ vD (rt2 nhip)⟩
    by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show "nsqnr (the (update rt dip (osn, kno, val, hops, nhip, {}) dip))
    < nsqnr (the (rt2 nhip dip))" .

```

qed

```
lemma dhops_le_hops_imp_update_strictly_fresher:
  assumes "dip ∈ vD(rt2 nhip)"
    and sqn: "sqn (rt2 nhip) dip = osn"
    and hop: "the (dhops (rt2 nhip) dip) ≤ hops"
    and **: "rt ≠ update rt dip (osn, kno, val, Suc hops, nhip, {})"
  shows "update rt dip (osn, kno, val, Suc hops, nhip, {}) ⊑dip rt2 nhip"
  unfolding rt_strictly_fresher_def
  proof (rule disjI2, rule conjI)
    from ** have "nsqn (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip = osn"
      by (rule nsqn_update_changed_kno_val)
    with ⟨dip ∈ vD(rt2 nhip)⟩
      have "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip)) = osn"
        by (simp add: kD_nsqn)
    also have "osn = sqn (rt2 nhip) dip" by (rule sqn [symmetric])
    also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip dip))"
      unfolding nsqnr_def using ⟨dip ∈ vD(rt2 nhip)⟩
      by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
    finally show "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip))
      = nsqnr (the (rt2 nhip dip))" .

  next
    have "the (dhops (rt2 nhip) dip) ≤ hops" by (rule hop)
    also have "hops < hops + 1" by simp
    also have "hops + 1 = the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)"
      using ** by simp
    finally have "the (dhops (rt2 nhip) dip)
      < the (dhops (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)" .
    thus "π5 (the (rt2 nhip dip)) < π5 (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip))"
      using ⟨dip ∈ vD(rt2 nhip)⟩ by (simp add: proj5_eq_dhops)
  qed
qed
```

```
lemma nsqn_invalidate:
  assumes "dip ∈ kD(rt)"
    and "∀ip ∈ dom(dests). ip ∈ vD(rt) ∧ the (dests ip) = inc (sqn rt ip)"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
  proof -
    from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp

    from assms have "rt ≈dip invalidate rt dests"
      by (rule rt_fresh_as_inc_invalidate)
    with ⟨dip ∈ kD(rt)⟩ ⟨dip ∈ kD(invalidate rt dests)⟩ show ?thesis
      by (simp add: kD_nsqn del: invalidate_kD_inv)
        (erule(2) rt_fresh_as_nsqnr)
  qed
qed
```

end

## 4.6 Invariant proofs on individual processes

```
theory D_Seq_Invariants
  imports "../.../AWN/Invariants" D_Aodv D_Aodv_Data D_Aodv_Predicates D_Fresher
begin
```

The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

```
lemma sequence_number_increases:
  "paodv i ⊨A onll ΓAODV (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
  by inv_cterms
```

```
lemma sequence_number_one_or_bigger:
  "paodv i ⊨ onl ΓAODV (λ(ξ, _). 1 ≤ sn ξ)"
  by (rule onll_step_to_invariantI [OF sequence_number_increases])
```

(auto simp:  $\sigma_{AODV\_def}$ )

We can get rid of the onl/onll if desired...

lemma sequence\_number\_increases':

"paadv i  $\models_A (\lambda((\xi, \_), \_, (\xi', \_)). sn \xi \leq sn \xi')$ "

by (rule step\_invariant\_weakenE [OF sequence\_number\_increases]) (auto dest!: onllD)

lemma sequence\_number\_one\_or\_bigger':

"paadv i  $\models (\lambda(\xi, \_). 1 \leq sn \xi)$ "

by (rule invariant\_weakenE [OF sequence\_number\_one\_or\_bigger]) auto

lemma sip\_in\_kD:

"paadv i  $\models onl \Gamma_{AODV} (\lambda(\xi, l). l \in (\{PAadv-:7\} \cup \{PAadv-:5\} \cup \{PRrep-:0..PRrep-:1\} \cup \{PRreq-:0..PRreq-:3\}) \longrightarrow sip \xi \in kD (rt \xi))"$

by inv\_cterms

lemma rrep\_1\_update\_changes:

"paadv i  $\models onl \Gamma_{AODV} (\lambda(\xi, l). (l = PRrep-:1 \longrightarrow$

$rt \xi \neq update (rt \xi) (dip \xi) (dsn \xi, kno, val, hops \xi + 1, sip \xi, \{ \}))"$

by inv\_cterms

lemma addpreRT\_partly\_welldefined:

"paadv i  $\models$

$onl \Gamma_{AODV} (\lambda(\xi, l). (l \in \{PRreq-:18..PRreq-:20\} \cup \{PRrep-:2..PRrep-:6\} \longrightarrow dip \xi \in kD (rt \xi)) \wedge (l \in \{PRreq-:3..PRreq-:19\} \longrightarrow oip \xi \in kD (rt \xi)))"$

by inv\_cterms

Proposition 7.38

lemma includes\_nhip:

"paadv i  $\models onl \Gamma_{AODV} (\lambda(\xi, l). \forall dip \in kD (rt \xi). the (nhop (rt \xi) dip) \in kD (rt \xi))"$

proof -

{ fix ip and  $\xi \xi' :: state$

assume " $\forall dip \in kD (rt \xi). the (nhop (rt \xi) dip) \in kD (rt \xi)$ "

and " $\xi' = \xi (|rt := update (rt \xi) ip (0, unk, val, Suc 0, ip, \{ \}))"$

hence " $\forall dip \in kD (rt \xi).$

$the (nhop (update (rt \xi) ip (0, unk, val, Suc 0, ip, \{ \})) dip) = ip$

$\vee the (nhop (update (rt \xi) ip (0, unk, val, Suc 0, ip, \{ \})) dip) \in kD (rt \xi)"$

by clarsimp (metis nhop\_update\_unk\_val update\_another)

} note one\_hop = this

{ fix ip sip sn hops and  $\xi \xi' :: state$

assume " $\forall dip \in kD (rt \xi). the (nhop (rt \xi) dip) \in kD (rt \xi)$ "

and " $\xi' = \xi (|rt := update (rt \xi) ip (sn, kno, val, Suc hops, sip, \{ \}))"$

and " $sip \in kD (rt \xi)"$

hence " $(the (nhop (update (rt \xi) ip (sn, kno, val, Suc hops, sip, \{ \})) ip) = ip$

$\vee the (nhop (update (rt \xi) ip (sn, kno, val, Suc hops, sip, \{ \})) ip) \in kD (rt \xi))$

$\wedge (\forall dip \in kD (rt \xi).$

$the (nhop (update (rt \xi) ip (sn, kno, val, Suc hops, sip, \{ \})) dip) = ip$

$\vee the (nhop (update (rt \xi) ip (sn, kno, val, Suc hops, sip, \{ \})) dip) \in kD (rt \xi))"$

by (metis kD\_update\_unchanged nhop\_update\_changed update\_another)

} note nhip\_is\_sip = this

show ?thesis

by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf sip\_in\_kD]

onl\_invariant\_sterms [OF aadv\_wf addpreRT\_partly\_welldefined]

solve: one\_hop nhip\_is\_sip)

qed

Proposition 7.22: needed in Proposition 7.4

lemma addpreRT\_welldefined:

"paadv i  $\models onl \Gamma_{AODV} (\lambda(\xi, l). (l \in \{PRreq-:18..PRreq-:20\} \longrightarrow dip \xi \in kD (rt \xi)) \wedge$

$(l = PRreq-:19 \longrightarrow oip \xi \in kD (rt \xi)) \wedge$

$(l = PRrep-:5 \longrightarrow dip \xi \in kD (rt \xi)) \wedge$

$(l = PRrep-:6 \longrightarrow (the (nhop (rt \xi) (dip \xi))) \in kD (rt \xi)))"$

(is " $\_ \models onl \Gamma_{AODV} ?P"$ )

unfolding invariant\_def

```

proof
  fix s
  assume "s ∈ reachable (paadv i) TT"
  then obtain ξ p where "s = (ξ, p)"
    and "(ξ, p) ∈ reachable (paadv i) TT"
  by (metis PairE)
  have "onl ΓAODV ?P (ξ, p)"
  proof (rule onlI)
    fix l
    assume "l ∈ labels ΓAODV p"
    with ⟨(ξ, p) ∈ reachable (paadv i) TT⟩
      have I1: "l ∈ {PRreq-:18..PRreq-:20} → dip ξ ∈ kD(rt ξ)"
      and I2: "l = PRreq-:19 → oip ξ ∈ kD(rt ξ)"
      and I3: "l ∈ {PRrep-:2..PRrep-:6} → dip ξ ∈ kD(rt ξ)"
      by (auto dest!: invariantD [OF addpreRT_partly_welldefined])
    moreover from ⟨(ξ, p) ∈ reachable (paadv i) TT⟩ ⟨l ∈ labels ΓAODV p⟩ and I3
      have "l = PRrep-:6 → (the (nhop (rt ξ) (dip ξ))) ∈ kD(rt ξ)"
      by (auto dest!: invariantD [OF includes_nhip])
    ultimately show "?P (ξ, l)"
      by simp
  qed
  with ⟨s = (ξ, p)⟩ show "onl ΓAODV ?P s"
  by simp
qed

```

Proposition 7.4

```

lemma known_destinations_increase:
  "paadv i ⊨A onll ΓAODV (λ((ξ, _), _, (ξ', _)). kD (rt ξ) ⊆ kD (rt ξ'))"
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf addpreRT_welldefined]
      simp add: subset_insertI)

```

Proposition 7.5

```

lemma rreqs_increase:
  "paadv i ⊨A onll ΓAODV (λ((ξ, _), _, (ξ', _)). rreqs ξ ⊆ rreqs ξ')"
  by (inv_cterms simp add: subset_insertI)

```

lemma dests\_bigger\_than\_sqn:

```

"paadv i ⊨ onl ΓAODV (λ(ξ, l). l ∈ {PAadv-:15..PAadv-:19}
  ∪ {PPkt-:7..PPkt-:11}
  ∪ {PRreq-:11..PRreq-:15}
  ∪ {PRreq-:24..PRreq-:28}
  ∪ {PRrep-:10..PRrep-:14}
  ∪ {PRerr-:1..PRerr-:5}
  → (∀ip∈dom(dests ξ). ip∈kD(rt ξ) ∧ sqn (rt ξ) ip ≤ the (dests ξ ip)))"

```

proof -

```

have sqninv:
  "∧dests rt rsn ip.
  [ ∀ip∈dom(dests). ip∈kD(rt) ∧ sqn rt ip ≤ the (dests ip); dests ip = Some rsn ]
  ⇒ sqn (invalidate rt dests) ip ≤ rsn"
  by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto
have indests:
  "∧dests rt rsn ip.
  [ ∀ip∈dom(dests). ip∈kD(rt) ∧ sqn rt ip ≤ the (dests ip); dests ip = Some rsn ]
  ⇒ ip∈kD(rt) ∧ sqn rt ip ≤ rsn"
  by (metis domI option.sel)
show ?thesis
  by inv_cterms
  (clarsimp split: split_if_asm option.split_asm
  elim!: sqninv indests)+

```

qed

Proposition 7.6

```

lemma sqns_increase:
  "paadv i ⊨A onll ΓAODV (λ((ξ, _), _, (ξ', _)). ∀ip. sqn (rt ξ) ip ≤ sqn (rt ξ') ip)"

```



```

proof -
  { fix  $\xi$  :: state
    assume *: " $\forall ip \in \text{dom}(\text{dests } \xi). ip \in \text{kD } (rt \ \xi) \wedge \text{sqn } (rt \ \xi) \ ip \leq \text{the } (\text{dests } \xi \ ip)$ "
    have " $\forall ip. \text{sqn } (rt \ \xi) \ ip \leq \text{sqn } (\text{invalidate } (rt \ \xi) \ (\text{dests } \xi)) \ ip$ "
    proof
      fix ip
      from * have " $ip \notin \text{dom}(\text{dests } \xi) \vee \text{sqn } (rt \ \xi) \ ip \leq \text{the } (\text{dests } \xi \ ip)$ " by simp
      thus " $\text{sqn } (rt \ \xi) \ ip \leq \text{sqn } (\text{invalidate } (rt \ \xi) \ (\text{dests } \xi)) \ ip$ "
        by (metis domI invalidate_sqn option.sel)
    qed
  } note solve_invalidate = this
  show ?thesis
    by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf addpreRT_welldefined]
        onl_invariant_sterms [OF aadv_wf dests_bigger_than_sqn]
        simp add: solve_invalidate)
qed

```

Proposition 7.7

```

lemma ip_constant:
  "paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, \_). ip \ \xi = i)$ "
  by (inv_cterms simp add:  $\sigma_{AODV\_def}$ )

```

Proposition 7.8

```

lemma sender_ip_valid':
  "paadv i  $\models_A$  onll  $\Gamma_{AODV} (\lambda((\xi, \_), a, \_). \text{anycast } (\lambda m. \text{not\_Pkt } m \longrightarrow \text{msg\_sender } m = ip \ \xi) \ a)$ "
  by inv_cterms

```

```

lemma sender_ip_valid:
  "paadv i  $\models_A$  onll  $\Gamma_{AODV} (\lambda((\xi, \_), a, \_). \text{anycast } (\lambda m. \text{not\_Pkt } m \longrightarrow \text{msg\_sender } m = i) \ a)$ "
  by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid'])
    (auto dest!: onlD onllD)

```

```

lemma received_msg_inv:
  "paadv i  $\models$  (recvm $\text{sg } P \rightarrow$ ) onl  $\Gamma_{AODV} (\lambda(\xi, l). l \in \{PAadv-:1\} \longrightarrow P \ (\text{msg } \xi))$ "
  by inv_cterms

```

Proposition 7.9

```

lemma sip_not_ip':
  "paadv i  $\models$  (recvm $\text{sg } (\lambda m. \text{not\_Pkt } m \longrightarrow \text{msg\_sender } m \neq i) \rightarrow)$  onl  $\Gamma_{AODV} (\lambda(\xi, \_). \text{sip } \xi \neq ip \ \xi)$ "
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]
        onl_invariant_sterms [OF aadv_wf ip_constant [THEN invariant_restrict_inD]]
        simp add: clear_locals_sip_not_ip') clarsimp+

```

```

lemma sip_not_ip:
  "paadv i  $\models$  (recvm $\text{sg } (\lambda m. \text{not\_Pkt } m \longrightarrow \text{msg\_sender } m \neq i) \rightarrow)$  onl  $\Gamma_{AODV} (\lambda(\xi, \_). \text{sip } \xi \neq i)$ "
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]
        onl_invariant_sterms [OF aadv_wf ip_constant [THEN invariant_restrict_inD]]
        simp add: clear_locals_sip_not_ip') clarsimp+

```

Neither `sip_not_ip'` nor `sip_not_ip` is needed to show loop freedom.

Proposition 7.10

```

lemma hop_count_positive:
  "paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, \_). \forall ip \in \text{kD } (rt \ \xi). \text{the } (\text{dhops } (rt \ \xi) \ ip) \geq 1)$ "
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf addpreRT_welldefined]) auto

```

```

lemma rreq_dip_in_vD_dip_eq_ip:
  "paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, l). (l \in \{PRreq-:18..PRreq-:21\} \longrightarrow \text{dip } \xi \in \text{vD}(rt \ \xi))
    \wedge (l \in \{PRreq-:6, PRreq-:7\} \longrightarrow \text{dip } \xi = ip \ \xi)
    \wedge (l \in \{PRreq-:17..PRreq-:21\} \longrightarrow \text{dip } \xi \neq ip \ \xi))$ "

```

```

proof (inv_cterms, elim conjE)
  fix l  $\xi$  pp p'
  assume " $(\xi, pp) \in \text{reachable } (\text{paadv } i) \ TT$ "

```

```

and "{PRreq-:19}[[λξ. ξ(rt := the (addpreRT (rt ξ) (oip ξ) {the (nhop (rt ξ) (dip ξ))})]] p'
  ∈ sterms ΓAODV pp"
and "l = PRreq-:19"
and "dip ξ ∈ vD (rt ξ)"
from this(1-3) have "oip ξ ∈ kD (rt ξ)"
by (auto dest: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined, where l="PRreq-:19"])
with ⟨dip ξ ∈ vD (rt ξ)⟩
show "dip ξ ∈ vD (the (addpreRT (rt ξ) (oip ξ) {the (nhop (rt ξ) (dip ξ))}))" by simp
qed

```

Proposition 7.11

lemma anycast\_msg\_zhops:

```

"∧rreqid dip dsn dsk oip osn sip.
  paodv i ⊨A onl ΓAODV (λ(., a, _). anycast msg_zhops a)"
proof (inv_cterms inv add:
  onl_invariant_sterms [OF aodv_wf rreq_dip_in_vD_dip_eq_ip [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN
    invariant_restrict_inD]],
  elim conjE)
  fix l ξ a pp p' pp'
  assume "(ξ, pp) ∈ reachable (paodv i) TT"
  and "{PRreq-:20}unicast(λξ. the (nhop (rt ξ) (oip ξ)),
    λξ. Rrep (the (dhops (rt ξ) (dip ξ))) (dip ξ) (sqn (rt ξ) (dip ξ)) (oip ξ) (ip ξ)).
    p' ▷ pp' ∈ sterms ΓAODV pp"
  and "l = PRreq-:20"
  and "a = unicast (the (nhop (rt ξ) (oip ξ)))
    (Rrep (the (dhops (rt ξ) (dip ξ))) (dip ξ) (sqn (rt ξ) (dip ξ)) (oip ξ) (ip ξ))"
  and *: "∀ip∈kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) ip)"
  and "dip ξ ∈ vD (rt ξ)"
  from ⟨dip ξ ∈ vD (rt ξ)⟩ have "dip ξ ∈ kD (rt ξ)"
  by (rule vD_id_gives_kD(1))
  with * have "Suc 0 ≤ the (dhops (rt ξ) (dip ξ))" ..
  thus "0 < the (dhops (rt ξ) (dip ξ))" by simp
qed

```

lemma hop\_count\_zero\_oip\_dip\_sip:

```

"paodv i ⊨ (recvmmsg msg_zhops →) onl ΓAODV (λ(ξ, l).
  (l ∈ {PAodv-:4..PAodv-:5} ∪ {PRreq-:n/n. True} →
    (hops ξ = 0 → oip ξ = sip ξ))
  ∧
  ((l ∈ {PAodv-:6..PAodv-:7} ∪ {PRrep-:n/n. True} →
    (hops ξ = 0 → dip ξ = sip ξ))))"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) auto

```

lemma osn\_rreq:

```

"paodv i ⊨ (recvmmsg rreq_rrep_sn →) onl ΓAODV (λ(ξ, l).
  l ∈ {PAodv-:4, PAodv-:5} ∪ {PRreq-:n/n. True} → 1 ≤ osn ξ)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

```

lemma osn\_rreq':

```

"paodv i ⊨ (recvmmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl ΓAODV (λ(ξ, l).
  l ∈ {PAodv-:4, PAodv-:5} ∪ {PRreq-:n/n. True} → 1 ≤ osn ξ)"
proof (rule invariant_weakenE [OF osn_rreq])
  fix a
  assume "recvmmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a"
  thus "recvmmsg rreq_rrep_sn a"
  by (cases a) simp_all
qed

```

lemma dsn\_rrep:

```

"paodv i ⊨ (recvmmsg rreq_rrep_sn →) onl ΓAODV (λ(ξ, l).
  l ∈ {PAodv-:6, PAodv-:7} ∪ {PRrep-:n/n. True} → 1 ≤ dsn ξ)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

```

lemma dsn\_rrep':

```
"paadv i  $\models$  (recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ .
   $1 \in \{PAadv-:6, PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow 1 \leq dsn\ \xi$ )"
```

proof (rule invariant\_weakenE [OF dsn\_rrep])

```
fix a
assume "recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ ) a"
thus "recvmsg rreq_rrep_sn a"
  by (cases a) simp_all
```

qed

lemma hop\_count\_zero\_oip\_dip\_sip':

```
"paadv i  $\models$  (recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ .
  ( $1 \in \{PAadv-:4..PAadv-:5\} \cup \{PRreq-:n/n. True\} \rightarrow$ 
    ( $hops\ \xi = 0 \rightarrow oip\ \xi = sip\ \xi$ ))
   $\wedge$ 
  ( $(1 \in \{PAadv-:6..PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow$ 
    ( $hops\ \xi = 0 \rightarrow dip\ \xi = sip\ \xi$ ))))"
```

proof (rule invariant\_weakenE [OF hop\_count\_zero\_oip\_dip\_sip])

```
fix a
assume "recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ ) a"
thus "recvmsg msg_zhops a"
  by (cases a) simp_all
```

qed

Proposition 7.12

lemma zero\_seq\_unk\_hops\_one':

```
"paadv i  $\models$  (recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_)$ .
   $\forall dip \in kD(rt\ \xi). (sqn\ (rt\ \xi)\ dip = 0 \rightarrow sqnf\ (rt\ \xi)\ dip = unk)$ 
   $\wedge (sqnf\ (rt\ \xi)\ dip = unk \rightarrow the\ (dhops\ (rt\ \xi)\ dip) = 1)$ 
   $\wedge (the\ (dhops\ (rt\ \xi)\ dip) = 1 \rightarrow the\ (nhop\ (rt\ \xi)\ dip) = dip))"$ 
```

proof -

```
{ fix dip and  $\xi :: state$  and P
  assume "sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip = 0"
  and all: " $\forall ip. sqn\ (rt\ \xi)\ ip \leq sqn\ (invalidate\ (rt\ \xi)\ (dests\ \xi))\ ip$ "
  and *: "sqn (rt  $\xi$ ) dip = 0  $\implies$  P  $\xi$  dip"
  have "P  $\xi$  dip"
```

proof -

```
from all have "sqn (rt  $\xi$ ) dip  $\leq$  sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip" ..
with  $\langle sqn\ (invalidate\ (rt\ \xi)\ (dests\ \xi))\ dip = 0 \rangle$  have "sqn (rt  $\xi$ ) dip = 0" by simp
thus "P  $\xi$  dip" by (rule *)
```

qed

```
} note sqn_invalidate_zero [elim!] = this
```

```
{ fix dsn hops :: nat and sip oip rt and ip dip :: ip
```

```
assume " $\forall dip \in kD(rt)$ .
  ( $sqn\ rt\ dip = 0 \rightarrow \pi_3(the\ (rt\ dip)) = unk$ )  $\wedge$ 
  ( $\pi_3(the\ (rt\ dip)) = unk \rightarrow the\ (dhops\ rt\ dip) = Suc\ 0$ )  $\wedge$ 
  ( $the\ (dhops\ rt\ dip) = Suc\ 0 \rightarrow the\ (nhop\ rt\ dip) = dip$ )"
  and "hops = 0  $\rightarrow sip = dip$ "
  and "Suc 0  $\leq$  dsn"
  and "ip  $\neq$  dip  $\rightarrow ip \in kD(rt)$ "
```

```
hence "the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = Suc 0  $\rightarrow$ 
  the (nhop (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = ip"
```

```
by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
```

```
} note prreq_ok1 [simp] = this
```

```
{ fix ip dsn hops sip oip rt dip
```

```
assume " $\forall dip \in kD(rt)$ .
  ( $sqn\ rt\ dip = 0 \rightarrow \pi_3(the\ (rt\ dip)) = unk$ )  $\wedge$ 
  ( $\pi_3(the\ (rt\ dip)) = unk \rightarrow the\ (dhops\ rt\ dip) = Suc\ 0$ )  $\wedge$ 
  ( $the\ (dhops\ rt\ dip) = Suc\ 0 \rightarrow the\ (nhop\ rt\ dip) = dip$ )"
  and "Suc 0  $\leq$  dsn"
  and "ip  $\neq$  dip  $\rightarrow ip \in kD(rt)$ "
```

```
hence " $\pi_3(the\ (update\ rt\ dip\ (dsn,\ kno,\ val,\ Suc\ hops,\ sip,\ \{ \})\ ip)) = unk \rightarrow$ 
```

```

    the (dhops (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = Suc 0"
  by - (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)
} note prreq_ok2 [simp] = this

{ fix ip dsn hops sip oip rt dip
  assume "∀dip∈kD(rt).
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
    and "Suc 0 ≤ dsn"
    and "ip ≠ dip → ip∈kD(rt)"
  hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip = 0 →
    π3 (the (update rt dip (dsn, kno, val, Suc hops, sip, {})) ip) = unk"
    by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
} note prreq_ok3 [simp] = this

{ fix rt sip
  assume "∀dip∈kD rt.
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
  hence "∀dip∈kD rt.
    (sqn (update rt sip (0, unk, val, Suc 0, sip, {})) dip = 0 →
    π3(the (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = unk)
    ∧ (π3(the (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = unk →
    the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = Suc 0)
    ∧ (the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = Suc 0 →
    the (nhop (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = dip)"
    by - (rule update_cases, simp_all add: sqn_def sqnf_def)
} note prreq_ok4 [simp] = this

have prreq_ok5 [simp]: "∧sip rt.
  π3(the (update rt sip (0, unk, val, Suc 0, sip, {})) sip) = unk →
  the (dhops (update rt sip (0, unk, val, Suc 0, sip, {})) sip) = Suc 0"
  by (rule update_cases) simp_all

have prreq_ok6 [simp]: "∧sip rt.
  sqn (update rt sip (0, unk, val, Suc 0, sip, {})) sip = 0 →
  π3 (the (update rt sip (0, unk, val, Suc 0, sip, {})) sip) = unk"
  by (rule update_cases) simp_all

show ?thesis
  by (inv_terms inv add: onl_invariant_sterms_TT [OF aadv_wf addpreRT_welldefined]
    onl_invariant_sterms [OF aadv_wf hop_count_zero_oip_dip_sip']
    seq_step_invariant_sterms_TT [OF sqns_increase aadv_wf aadv_trans]
    onl_invariant_sterms [OF aadv_wf osn_rreq']
    onl_invariant_sterms [OF aadv_wf dsn_rrep']) clarsimp+

qed

lemma zero_seq_unk_hops_one:
  "paadv i ⊨ (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl ΓAODV (λ(ξ, _).
    ∀dip∈kD(rt ξ). (sqn (rt ξ) dip = 0 → (sqnf (rt ξ) dip = unk
    ∧ the (dhops (rt ξ) dip) = 1
    ∧ the (nhop (rt ξ) dip) = dip)))"
  by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

lemma kD_unk_or_atleast_one:
  "paadv i ⊨ (recvmsg rreq_rrep_sn →) onl ΓAODV (λ(ξ, l).
    ∀dip∈kD(rt ξ). π3(the (rt ξ dip)) = unk ∨ 1 ≤ π2(the (rt ξ dip)))"

proof -
  { fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
    assume "dsk1 = unk ∨ Suc 0 ≤ dsn2"
    hence "π3(the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1, pre1) sip)) = unk
    ∨ Suc 0 ≤ sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2)) sip"
  }

```

```

    unfolding update_def by (cases "dsk1 =unk") (clarsimp split: option.split)+
  } note fromsip [simp] = this

{ fix dip sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
  assume allkd: " $\forall dip \in kD(rt). \pi_3(\text{the } (rt \text{ dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } rt \text{ dip}$ "
    and **: " $\text{dsk1} = \text{unk} \vee \text{Suc } 0 \leq \text{dsn2}$ "
  have " $\forall dip \in kD(rt). \pi_3(\text{the } (\text{update } rt \text{ sip } (\text{dsn1}, \text{dsk1}, \text{flag1}, \text{hops1}, \text{nhip1}, \text{pre1}) \text{ dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } (\text{update } rt \text{ sip } (\text{dsn2}, \text{dsk2}, \text{flag2}, \text{hops2}, \text{nhip2}, \text{pre2})) \text{ dip}$ "
    (is " $\forall dip \in kD(rt). ?prop \text{ dip}$ ")
  proof
    fix dip
    assume "dip  $\in kD(rt)$ "
    thus "?prop dip"
    proof (cases "dip = sip")
      assume "dip = sip"
      with ** show ?thesis
      by simp
    next
      assume "dip  $\neq$  sip"
      with  $\langle dip \in kD(rt) \rangle$  allkd show ?thesis
      by simp
    qed
  qed
} note solve_update [simp] = this

{ fix dip rt dests
  assume *: " $\forall ip \in \text{dom}(\text{dests}). ip \in kD(rt) \wedge \text{sqn } rt \text{ ip} \leq \text{the } (\text{dests } ip)$ "
    and **: " $\forall ip \in kD(rt). \pi_3(\text{the } (rt \text{ ip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } rt \text{ ip}$ "
  have " $\forall dip \in kD(rt). \pi_3(\text{the } (rt \text{ dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } (\text{invalidate } rt \text{ dests}) \text{ dip}$ "
  proof
    fix dip
    assume "dip  $\in kD(rt)$ "
    with ** have " $\pi_3(\text{the } (rt \text{ dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } rt \text{ dip}$ " ..
    thus " $\pi_3(\text{the } (rt \text{ dip})) = \text{unk} \vee \text{Suc } 0 \leq \text{sqn } (\text{invalidate } rt \text{ dests}) \text{ dip}$ "
    proof
      assume " $\pi_3(\text{the } (rt \text{ dip})) = \text{unk}$ " thus ?thesis ..
    next
      assume "Suc 0  $\leq$  sqn rt dip"
      have "Suc 0  $\leq$  sqn (invalidate rt dests) dip"
      proof (cases "dip  $\in \text{dom}(\text{dests})$ ")
        assume "dip  $\in \text{dom}(\text{dests})$ "
        with * have "sqn rt dip  $\leq$  the (dests dip)" by simp
        with  $\langle \text{Suc } 0 \leq \text{sqn } rt \text{ dip} \rangle$  have "Suc 0  $\leq$  the (dests dip)" by simp
        with  $\langle dip \in \text{dom}(\text{dests}) \rangle \langle dip \in kD(rt) \rangle$  [THEN kD_Some] show ?thesis
          unfolding invalidate_def sqn_def by auto
      next
        assume "dip  $\notin \text{dom}(\text{dests})$ "
        with  $\langle \text{Suc } 0 \leq \text{sqn } rt \text{ dip} \rangle \langle dip \in kD(rt) \rangle$  [THEN kD_Some] show ?thesis
          unfolding invalidate_def sqn_def by auto
      qed
    qed
    thus ?thesis by (rule disjI2)
  qed
} note solve_invalidate [simp] = this

show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
      onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn
                              [THEN invariant_restrict_inD]]
      onl_invariant_sterms [OF aodv_wf osn_rreq]
      onl_invariant_sterms [OF aodv_wf dsn_rrep]
      simp add: proj3_inv proj2_eq_sqn)

```

qed

Proposition 7.13

lemma rreq\_rrep\_sn\_any\_step\_invariant:

"paadv i  $\models_A$  (recvmsg rreq\_rrep\_sn  $\rightarrow$ ) onll  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_)$ . anycast rreq\_rrep\_sn a)"

proof -

have sqnf\_kno: "paadv i  $\models$  onll  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ .

( $1 \in \{\text{PRreq-:18..PRreq-:20}\} \rightarrow \text{sqnf}(\text{rt } \xi) (\text{dip } \xi) = \text{kno}))$ "

by (inv\_cterms inv add: onll\_invariant\_sterms\_TT [OF aadv\_wf addpreRT\_welldefined])

show ?thesis

by (inv\_cterms inv add: onll\_invariant\_sterms\_TT [OF aadv\_wf addpreRT\_welldefined]

onll\_invariant\_sterms [OF aadv\_wf sequence\_number\_one\_or\_bigger

[THEN invariant\_restrict\_inD]]

onll\_invariant\_sterms [OF aadv\_wf kD\_unk\_or\_atleast\_one]

onll\_invariant\_sterms\_TT [OF aadv\_wf sqnf\_kno]

onll\_invariant\_sterms [OF aadv\_wf osn\_rreq]

onll\_invariant\_sterms [OF aadv\_wf dsn\_rrep])

(auto simp: proj2\_eq\_sqn)

qed

Proposition 7.14

lemma rreq\_rrep\_fresh\_any\_step\_invariant:

"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast (rreq\_rrep\_fresh (rt  $\xi$ )) a)"

proof -

have rreq\_oip: "paadv i  $\models$  onll  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ .

( $1 \in \{\text{PRreq-:3..PRreq-:9}\} \cup \{\text{PRreq-:17, PRreq-:30, PRreq-:32}\}$

$\rightarrow \text{oip } \xi \in \text{kD}(\text{rt } \xi)$

$\wedge (\text{sqn}(\text{rt } \xi) (\text{oip } \xi) > (\text{osn } \xi)$

$\vee (\text{sqn}(\text{rt } \xi) (\text{oip } \xi) = (\text{osn } \xi)$

$\wedge \text{the}(\text{dhops}(\text{rt } \xi) (\text{oip } \xi)) \leq \text{Suc}(\text{hops } \xi)$

$\wedge \text{the}(\text{flag}(\text{rt } \xi) (\text{oip } \xi)) = \text{val}))$ "

proof inv\_cterms

fix l  $\xi$  l' pp p'

assume " $(\xi, \text{pp}) \in \text{reachable}(\text{paadv } i)$  TT"

and " $\{\text{PRreq-:2}\} \ll \lambda \xi. \xi(\text{rt} :=$

$\text{update}(\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops } \xi), \text{sip } \xi, \{\})$ )" p'  $\in \text{sterms } \Gamma_{AODV}$  pp"

and "l' = PRreq-:3"

show "osn  $\xi <$  sqn ( $\text{update}(\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops } \xi), \text{sip } \xi, \{\})$ ) (oip  $\xi$ )

$\vee (\text{sqn}(\text{update}(\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops } \xi), \text{sip } \xi, \{\})) (\text{oip } \xi) = \text{osn } \xi$

$\wedge \text{the}(\text{dhops}(\text{update}(\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops } \xi), \text{sip } \xi, \{\})) (\text{oip } \xi))$

$\leq \text{Suc}(\text{hops } \xi)$

$\wedge \text{the}(\text{flag}(\text{update}(\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops } \xi), \text{sip } \xi, \{\})) (\text{oip } \xi))$

$= \text{val})$ "

unfolding update\_def by (clarsimp split: option.split)

(metis linorder\_neqE\_nat not\_less)

qed

have rrep\_prrep: "paadv i  $\models$  onll  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ .

( $1 \in \{\text{PRrep-:2..PRrep-:7}\} \rightarrow (\text{dip } \xi \in \text{kD}(\text{rt } \xi)$

$\wedge \text{sqn}(\text{rt } \xi) (\text{dip } \xi) = \text{dsn } \xi$

$\wedge \text{the}(\text{dhops}(\text{rt } \xi) (\text{dip } \xi)) = \text{Suc}(\text{hops } \xi)$

$\wedge \text{the}(\text{flag}(\text{rt } \xi) (\text{dip } \xi)) = \text{val}$

$\wedge \text{the}(\text{nhop}(\text{rt } \xi) (\text{dip } \xi)) \in \text{kD}(\text{rt } \xi)))$ "

by (inv\_cterms inv add: onll\_invariant\_sterms [OF aadv\_wf rrep\_1\_update\_changes]

onll\_invariant\_sterms [OF aadv\_wf sip\_in\_kD])

have rreq\_oip\_kD: "paadv i  $\models$  onll  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ . ( $1 \in \{\text{PRreq-:3..PRreq-:28}\} \rightarrow \text{oip } \xi \in \text{kD}(\text{rt } \xi))$ )"

by (inv\_cterms inv add: onll\_invariant\_sterms\_TT [OF aadv\_wf addpreRT\_welldefined])

have rreq\_dip\_kD\_oip\_sqn: "paadv i  $\models$  onll  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ .

( $1 \in \{\text{PRreq-:18..PRreq-:21}\}$

$\rightarrow (\text{dip } \xi \in \text{kD}(\text{rt } \xi)$

$\wedge (\text{sqn}(\text{rt } \xi) (\text{oip } \xi) > (\text{osn } \xi)$

$\vee (\text{sqn}(\text{rt } \xi) (\text{oip } \xi) = (\text{osn } \xi)$

$\wedge \text{the}(\text{dhops}(\text{rt } \xi) (\text{oip } \xi)) \leq \text{Suc}(\text{hops } \xi)$

$\wedge \text{the}(\text{flag}(\text{rt } \xi) (\text{oip } \xi)) = \text{val}))$ )"

```
by(inv_cterms inv add: onl_invariant_sterms [OF aadv_wf rreq_oip]
    onl_invariant_sterms [OF aadv_wf addpreRT_welldefined])
```

show ?thesis

```
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf rreq_oip]
    onl_invariant_sterms [OF aadv_wf rreq_dip_in_vD_dip_eq_ip]
    onl_invariant_sterms [OF aadv_wf rrep_prrep]
    onl_invariant_sterms [OF aadv_wf rreq_oip_kD]
    onl_invariant_sterms [OF aadv_wf rreq_dip_kD_oip_sqn])
```

qed

Proposition 7.15

lemma rerr\_invalid\_any\_step\_invariant:

"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast (rerr\_invalid (rt  $\xi$ )) a)"

proof -

```
have dests_inv: "paadv i  $\models$ 
    onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ . (l  $\in$  {PAadv-:15, PPkt-:7, PRreq-:11,
        PRreq-:24, PRrep-:10, PRerr-:1}
         $\rightarrow$  ( $\forall ip \in \text{dom}(\text{dests } \xi)$ . ip  $\in$  vD(rt  $\xi$ )))
     $\wedge$  (l  $\in$  {PAadv-:16..PAadv-:19}
         $\cup$  {PPkt-:8..PPkt-:11}
         $\cup$  {PRreq-:12..PRreq-:15}
         $\cup$  {PRreq-:25..PRreq-:28}
         $\cup$  {PRrep-:11..PRrep-:14}
         $\cup$  {PRerr-:2..PRerr-:5}  $\rightarrow$  ( $\forall ip \in \text{dom}(\text{dests } \xi)$ . ip  $\in$  iD(rt  $\xi$ )
             $\wedge$  the (dests  $\xi$  ip) = sqn (rt  $\xi$ ) ip))
     $\wedge$  (l = PPkt-:14  $\rightarrow$  dip  $\xi \in$  iD(rt  $\xi$ )))"
```

by inv\_cterms (clarsimp split: split\_if\_asm option.split\_asm simp: domIff)+

show ?thesis

```
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf dests_inv])
```

qed

Proposition 7.16

Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs ( $rip$ ,  $rsn$ ) is assigned to the variable  $dests$  of type  $ip \rightarrow sqn$ , or to the first argument of the function  $rerr$ , this set is a partial function, i.e., there is at most one entry ( $rip$ ,  $rsn$ ) for each destination  $rip$ : guaranteed by type.

lemma dests\_vD\_inc\_sqn:

```
"paadv i  $\models$ 
    onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ . (l  $\in$  {PAadv-:15, PPkt-:7, PRreq-:11, PRreq-:24, PRrep-:10}
         $\rightarrow$  ( $\forall ip \in \text{dom}(\text{dests } \xi)$ . ip  $\in$  vD(rt  $\xi$ )  $\wedge$  the (dests  $\xi$  ip) = inc (sqn (rt  $\xi$ ) ip)))
     $\wedge$  (l = PRerr-:1
         $\rightarrow$  ( $\forall ip \in \text{dom}(\text{dests } \xi)$ . ip  $\in$  vD(rt  $\xi$ )  $\wedge$  the (dests  $\xi$  ip) > sqn (rt  $\xi$ ) ip)))"
```

by inv\_cterms (clarsimp split: split\_if\_asm option.split\_asm)+

Proposition 7.27

lemma route\_tables\_fresher:

```
"paadv i  $\models_A$  (recvmgs rreq_rrep_sn  $\rightarrow$ ) onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_))$ .
     $\forall dip \in kD(\text{rt } \xi)$ . rt  $\xi \sqsubseteq_{dip}$  rt  $\xi'$ )"
```

proof (inv\_cterms inv add:

```
    onl_invariant_sterms [OF aadv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
    onl_invariant_sterms [OF aadv_wf hop_count_positive [THEN invariant_restrict_inD]]
    onl_invariant_sterms [OF aadv_wf osn_rreq]
    onl_invariant_sterms [OF aadv_wf dsn_rrep]
    onl_invariant_sterms [OF aadv_wf addpreRT_welldefined [THEN invariant_restrict_inD]])
```

```

fix  $\xi$  pp p'
assume "( $\xi$ , pp)  $\in$  reachable (paadv i) (recvmmsg rreq_rrep_sn)"
  and "{PRreq-:2} $\llbracket$  $\lambda\xi$ .  $\xi$ (|rt := update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , { $\xi$ )})) $\rrbracket$ "
  and "p'  $\in$  sterms  $\Gamma_{AODV}$  pp"
  and "Suc 0  $\leq$  osn  $\xi$ "
  and *: " $\forall ip \in kD$  (rt  $\xi$ ). Suc 0  $\leq$  the (dhops (rt  $\xi$ ) ip)"
show " $\forall ip \in kD$  (rt  $\xi$ ). rt  $\xi$   $\sqsubseteq_{ip}$  update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , { $\xi$ })"
proof
  fix ip
  assume "ip  $\in kD$  (rt  $\xi$ )"
  moreover with * have "1  $\leq$  the (dhops (rt  $\xi$ ) ip)" by simp
  moreover from  $\langle$ Suc 0  $\leq$  osn  $\xi$  $\rangle$ 
    have "update_arg_wf (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , { $\xi$ })" ..
  ultimately show "rt  $\xi$   $\sqsubseteq_{ip}$  update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , { $\xi$ })"
    by (rule rt_fresher_update)
qed
next
fix  $\xi$  pp p'
assume "( $\xi$ , pp)  $\in$  reachable (paadv i) (recvmmsg rreq_rrep_sn)"
  and "{PRrep-:1} $\llbracket$  $\lambda\xi$ .  $\xi$ (|rt := update (rt  $\xi$ ) (dip  $\xi$ ) (dsn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , { $\xi$ )})) $\rrbracket$ "
  and "p'  $\in$  sterms  $\Gamma_{AODV}$  pp"
  and "Suc 0  $\leq$  dsn  $\xi$ "
  and *: " $\forall ip \in kD$  (rt  $\xi$ ). Suc 0  $\leq$  the (dhops (rt  $\xi$ ) ip)"
show " $\forall ip \in kD$  (rt  $\xi$ ). rt  $\xi$   $\sqsubseteq_{ip}$  update (rt  $\xi$ ) (dip  $\xi$ ) (dsn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , { $\xi$ })"
proof
  fix ip
  assume "ip  $\in kD$  (rt  $\xi$ )"
  moreover with * have "1  $\leq$  the (dhops (rt  $\xi$ ) ip)" by simp
  moreover from  $\langle$ Suc 0  $\leq$  dsn  $\xi$  $\rangle$ 
    have "update_arg_wf (dsn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , { $\xi$ })" ..
  ultimately show "rt  $\xi$   $\sqsubseteq_{ip}$  update (rt  $\xi$ ) (dip  $\xi$ ) (dsn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ , { $\xi$ })"
    by (rule rt_fresher_update)
qed
qed
end

```

## 4.7 The quality increases predicate

```

theory D_Quality_Increases
imports D_Aadv_Predicates D_Fresher
begin

```

```

definition quality_increases :: "state  $\Rightarrow$  state  $\Rightarrow$  bool"
where "quality_increases  $\xi$   $\xi'$   $\equiv$  ( $\forall dip \in kD$ (rt  $\xi$ ). dip  $\in kD$ (rt  $\xi'$ )  $\wedge$  rt  $\xi$   $\sqsubseteq_{dip}$  rt  $\xi'$ )
   $\wedge$  ( $\forall dip$ . sqn (rt  $\xi$ ) dip  $\leq$  sqn (rt  $\xi'$ ) dip)"

```

```

lemma quality_increasesI [intro!]:
  assumes " $\wedge dip$ . dip  $\in kD$ (rt  $\xi$ )  $\implies$  dip  $\in kD$ (rt  $\xi'$ )"
  and " $\wedge dip$ .  $\llbracket$  dip  $\in kD$ (rt  $\xi$ ); dip  $\in kD$ (rt  $\xi'$ )  $\rrbracket \implies$  rt  $\xi$   $\sqsubseteq_{dip}$  rt  $\xi'$ "
  and " $\wedge dip$ . sqn (rt  $\xi$ ) dip  $\leq$  sqn (rt  $\xi'$ ) dip"
  shows "quality_increases  $\xi$   $\xi'$ "
  unfolding quality_increases_def using assms by clarsimp

```

```

lemma quality_increasesE [elim]:
  fixes dip
  assumes "quality_increases  $\xi$   $\xi'$ "
  and "dip  $\in kD$ (rt  $\xi$ )"
  and " $\llbracket$  dip  $\in kD$ (rt  $\xi'$ ); rt  $\xi$   $\sqsubseteq_{dip}$  rt  $\xi'$ ; sqn (rt  $\xi$ ) dip  $\leq$  sqn (rt  $\xi'$ ) dip  $\rrbracket \implies$  R dip  $\xi$   $\xi'$ "
  shows "R dip  $\xi$   $\xi'$ "
  using assms unfolding quality_increases_def by clarsimp

```

```

lemma quality_increases_rt_fresherD [dest]:
  fixes ip

```



```

assumes "quality_increases  $\xi \xi'$ "
  and "ip $\in$ kD(rt  $\xi$ )"
  shows "rt  $\xi \sqsubseteq_{ip}$  rt  $\xi'$ "
using assms by auto

lemma quality_increases_sqnE [elim]:
  fixes dip
  assumes "quality_increases  $\xi \xi'$ "
    and "sqn (rt  $\xi$ ) dip  $\leq$  sqn (rt  $\xi'$ ) dip  $\implies$  R dip  $\xi \xi'$ "
  shows "R dip  $\xi \xi'$ "
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_refl [intro, simp]: "quality_increases  $\xi \xi$ "
  by rule simp_all

lemma strictly_fresher_quality_increases_right [elim]:
  fixes  $\sigma \sigma'$  dip
  assumes "rt ( $\sigma$  i)  $\sqsubseteq_{dip}$  rt ( $\sigma$  nhip)"
    and qinc: "quality_increases ( $\sigma$  nhip) ( $\sigma'$  nhip)"
    and "dip $\in$ kD(rt ( $\sigma$  nhip))"
  shows "rt ( $\sigma$  i)  $\sqsubseteq_{dip}$  rt ( $\sigma'$  nhip)"
proof -
  from qinc have "rt ( $\sigma$  nhip)  $\sqsubseteq_{dip}$  rt ( $\sigma'$  nhip)" using (dip $\in$ kD(rt ( $\sigma$  nhip)))
  by auto
  with (rt ( $\sigma$  i)  $\sqsubseteq_{dip}$  rt ( $\sigma$  nhip)) show ?thesis ..
qed

lemma kD_quality_increases [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "quality_increases  $\xi \xi'$ "
  shows "i $\in$ kD(rt  $\xi'$ )"
using assms by auto

lemma kD_nsqn_quality_increases [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "quality_increases  $\xi \xi'$ "
  shows "i $\in$ kD(rt  $\xi'$ )  $\wedge$  nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi'$ ) i"
proof -
  from assms have "i $\in$ kD(rt  $\xi'$ )" ..
  moreover with assms have "rt  $\xi \sqsubseteq_i$  rt  $\xi'$ " by auto
  ultimately have "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi'$ ) i"
  using (i $\in$ kD(rt  $\xi$ )) by - (erule(2) rt_fresher_imp_nsqn_le)
  with (i $\in$ kD(rt  $\xi'$ )) show ?thesis ..
qed

lemma nsqn_quality_increases [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "quality_increases  $\xi \xi'$ "
  shows "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi'$ ) i"
using assms by (rule kD_nsqn_quality_increases [THEN conjunct2])

lemma kD_nsqn_quality_increases_trans [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "s  $\leq$  nsqn (rt  $\xi$ ) i"
    and "quality_increases  $\xi \xi'$ "
  shows "i $\in$ kD(rt  $\xi'$ )  $\wedge$  s  $\leq$  nsqn (rt  $\xi'$ ) i"
proof
  from (i $\in$ kD(rt  $\xi$ )) and (quality_increases  $\xi \xi'$ ) show "i $\in$ kD(rt  $\xi'$ )" ..
next
  from (i $\in$ kD(rt  $\xi$ )) and (quality_increases  $\xi \xi'$ ) have "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi'$ ) i" ..
  with (s  $\leq$  nsqn (rt  $\xi$ ) i) show "s  $\leq$  nsqn (rt  $\xi'$ ) i" by (rule le_trans)
qed

lemma nsqn_quality_increases_nsqn_lt_lt [elim]:

```

```

assumes "i∈kD(rt ξ)"
  and "quality_increases ξ ξ'"
  and "s < nsqn (rt ξ) i"
shows "s < nsqn (rt ξ') i"
proof -
  from assms(1-2) have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
  with (s < nsqn (rt ξ) i) show "s < nsqn (rt ξ') i" by simp
qed

```

```

lemma nsqn_quality_increases_dhops [elim]:
  assumes "i∈kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "nsqn (rt ξ) i = nsqn (rt ξ') i"
  shows "the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i)"
using assms unfolding quality_increases_def
by (clarsimp) (drule(1) bspec, clarsimp simp: rt_fresher_def2)

```

```

lemma nsqn_quality_increases_nsqn_eq_le [elim]:
  assumes "i∈kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "s = nsqn (rt ξ) i"
  shows "s < nsqn (rt ξ') i ∨ (s = nsqn (rt ξ') i ∧ the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i))"
using assms by (metis nat_less_le nsqn_quality_increases nsqn_quality_increases_dhops)

```

```

lemma quality_increases_rreq_rrep_props [elim]:
  fixes sn ip hops sip
  assumes qinc: "quality_increases (σ sip) (σ' sip)"
    and "1 ≤ sn"
    and *: "ip∈kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
           ∧ (nsqn (rt (σ sip)) ip = sn
              → (the (dhops (rt (σ sip)) ip) ≤ hops
                  ∨ the (flag (rt (σ sip)) ip) = inv))"
  shows "ip∈kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip
        ∧ (nsqn (rt (σ' sip)) ip = sn
           → (the (dhops (rt (σ' sip)) ip) ≤ hops
              ∨ the (flag (rt (σ' sip)) ip) = inv))"
  (is "_ ∧ ?nsqnafter")

```

```

proof -
  from * obtain "ip∈kD(rt (σ sip))" and "sn ≤ nsqn (rt (σ sip)) ip" by auto

```

```

  from ⟨quality_increases (σ sip) (σ' sip)⟩
  have "sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip" ..
  from ⟨quality_increases (σ sip) (σ' sip)⟩ and ⟨ip∈kD (rt (σ sip))⟩
  have "ip∈kD (rt (σ' sip))" ..

```

```

  from ⟨sn ≤ nsqn (rt (σ sip)) ip⟩ have ?nsqnafter

```

```

proof
  assume "sn < nsqn (rt (σ sip)) ip"
  also from ⟨ip∈kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  have "... ≤ nsqn (rt (σ' sip)) ip" ..
  finally have "sn < nsqn (rt (σ' sip)) ip" .
  thus ?thesis by simp

```

```

next

```

```

  assume "sn = nsqn (rt (σ sip)) ip"
  with ⟨ip∈kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  have "sn < nsqn (rt (σ' sip)) ip
        ∨ (sn = nsqn (rt (σ' sip)) ip
           ∧ the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip))" ..
  hence "sn < nsqn (rt (σ' sip)) ip
        ∨ (nsqn (rt (σ' sip)) ip = sn ∧ (the (dhops (rt (σ' sip)) ip) ≤ hops
                                           ∨ the (flag (rt (σ' sip)) ip) = inv))"

```

```

proof
  assume "sn < nsqn (rt (σ' sip)) ip" thus ?thesis ..
next

```

```

assume "sn = nsqn (rt (σ' sip)) ip
      ∧ the (dhops (rt (σ sip)) ip) ≥ the (dhops (rt (σ' sip)) ip)"
hence "sn = nsqn (rt (σ' sip)) ip"
      and "the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)" by auto

from * and ⟨sn = nsqn (rt (σ sip)) ip⟩ have "the (dhops (rt (σ sip)) ip) ≤ hops
      ∨ the (flag (rt (σ sip)) ip) = inv"

by simp
thus ?thesis
proof
  assume "the (dhops (rt (σ sip)) ip) ≤ hops"
  with ⟨the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)⟩
  have "the (dhops (rt (σ' sip)) ip) ≤ hops" by simp
  with ⟨sn = nsqn (rt (σ' sip)) ip⟩ show ?thesis by simp
next
  assume "the (flag (rt (σ sip)) ip) = inv"
  with ⟨ip ∈ kD(rt (σ sip))⟩ have "nsqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip - 1" ..

  with ⟨sn ≥ 1⟩ and ⟨sn = nsqn (rt (σ sip)) ip⟩
  have "sqn (rt (σ sip)) ip > 1" by simp

  from ⟨ip ∈ kD(rt (σ' sip))⟩ show ?thesis
  proof (rule vD_or_iD)
    assume "ip ∈ iD(rt (σ' sip))"
    hence "the (flag (rt (σ' sip)) ip) = inv" ..
    with ⟨sn = nsqn (rt (σ' sip)) ip⟩ show ?thesis
      by simp
  next
    assume "ip ∈ vD(rt (σ' sip))"
    hence "nsqn (rt (σ' sip)) ip = sqn (rt (σ' sip)) ip" ..
    with ⟨sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip⟩
    have "nsqn (rt (σ' sip)) ip ≥ sqn (rt (σ sip)) ip" by simp

    with ⟨sqn (rt (σ sip)) ip > 1⟩
    have "nsqn (rt (σ' sip)) ip > sqn (rt (σ sip)) ip - 1" by simp
    with ⟨nsqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip - 1⟩
    have "nsqn (rt (σ' sip)) ip > nsqn (rt (σ sip)) ip" by simp
    with ⟨sn = nsqn (rt (σ sip)) ip⟩ have "nsqn (rt (σ' sip)) ip > sn"
      by simp
    thus ?thesis ..
  qed
qed
qed
qed
thus ?thesis by (metis (mono_tags) le_cases not_le)
qed
with ⟨ip ∈ kD (rt (σ' sip))⟩ show "ip ∈ kD (rt (σ' sip)) ∧ ?nsqnafter" ..
qed

```

lemma quality\_increases\_rreq\_rrep\_props':

```

fixes sn ip hops sip
assumes "∀j. quality_increases (σ j) (σ' j)"
      and "1 ≤ sn"
      and *: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
      ∧ (nsqn (rt (σ sip)) ip = sn
      → (the (dhops (rt (σ sip)) ip) ≤ hops
      ∨ the (flag (rt (σ sip)) ip) = inv))"
shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip
      ∧ (nsqn (rt (σ' sip)) ip = sn
      → (the (dhops (rt (σ' sip)) ip) ≤ hops
      ∨ the (flag (rt (σ' sip)) ip) = inv))"

```

proof -

```

from assms(1) have "quality_increases (σ sip) (σ' sip)" ..
thus ?thesis using assms(2-3) by (rule quality_increases_rreq_rrep_props)

```

qed

lemma rteq\_quality\_increases:

assumes " $\forall j. j \neq i \longrightarrow \text{quality\_increases } (\sigma j) (\sigma' j)$ "

and " $\text{rt } (\sigma' i) = \text{rt } (\sigma i)$ "

shows " $\forall j. \text{quality\_increases } (\sigma j) (\sigma' j)$ "

using assms by clarsimp (metis order\_refl quality\_increasesI rt\_fresher\_refl)

definition msg\_fresh :: "(ip  $\Rightarrow$  state)  $\Rightarrow$  msg  $\Rightarrow$  bool"

where "msg\_fresh  $\sigma$  m  $\equiv$

case m of Rreq hops<sub>c</sub> \_ \_ \_ oipc osnc sipc \_  $\Rightarrow$  osnc  $\geq$  1  $\wedge$  (sipc  $\neq$  oipc  $\longrightarrow$   
oipc  $\in$  kD(rt ( $\sigma$  sipc))  $\wedge$  nsqn (rt ( $\sigma$  sipc)) oipc  $\geq$  osnc  
 $\wedge$  (nsqn (rt ( $\sigma$  sipc)) oipc = osnc  
 $\longrightarrow$  (hops  $\geq$  the (dhops (rt ( $\sigma$  sipc)) oipc)  
 $\vee$  the (flag (rt ( $\sigma$  sipc)) oipc) = inv)))  
| Rrep hops<sub>c</sub> dipc dsnc \_ sipc  $\Rightarrow$  dsnc  $\geq$  1  $\wedge$  (sipc  $\neq$  dipc  $\longrightarrow$   
dipc  $\in$  kD(rt ( $\sigma$  sipc))  $\wedge$  nsqn (rt ( $\sigma$  sipc)) dipc  $\geq$  dsnc  
 $\wedge$  (nsqn (rt ( $\sigma$  sipc)) dipc = dsnc  
 $\longrightarrow$  (hops  $\geq$  the (dhops (rt ( $\sigma$  sipc)) dipc)  
 $\vee$  the (flag (rt ( $\sigma$  sipc)) dipc) = inv)))  
| Rerr destsc sipc  $\Rightarrow$  ( $\forall$  ripc  $\in$  dom(destsc). (ripic  $\in$  kD(rt ( $\sigma$  sipc))  
 $\wedge$  the (destsc ripc) - 1  $\leq$  nsqn (rt ( $\sigma$  sipc)) ripc))  
| \_  $\Rightarrow$  True"

lemma msg\_fresh [simp]:

" $\wedge$ hops rreqid dip dsn dsk oip osn sip handled.

msg\_fresh  $\sigma$  (Rreq hops rreqid dip dsn dsk oip osn sip handled) =

(osn  $\geq$  1  $\wedge$  (sip  $\neq$  oip  $\longrightarrow$  oip  $\in$  kD(rt ( $\sigma$  sip))

$\wedge$  nsqn (rt ( $\sigma$  sip)) oip  $\geq$  osn

$\wedge$  (nsqn (rt ( $\sigma$  sip)) oip = osn

$\longrightarrow$  (hops  $\geq$  the (dhops (rt ( $\sigma$  sip)) oip)

$\vee$  the (flag (rt ( $\sigma$  sip)) oip) = inv)))"

" $\wedge$ hops dip dsn oip sip. msg\_fresh  $\sigma$  (Rrep hops dip dsn oip sip) =

(dsn  $\geq$  1  $\wedge$  (sip  $\neq$  dip  $\longrightarrow$  dip  $\in$  kD(rt ( $\sigma$  sip))

$\wedge$  nsqn (rt ( $\sigma$  sip)) dip  $\geq$  dsn

$\wedge$  (nsqn (rt ( $\sigma$  sip)) dip = dsn

$\longrightarrow$  (hops  $\geq$  the (dhops (rt ( $\sigma$  sip)) dip))

$\vee$  the (flag (rt ( $\sigma$  sip)) dip) = inv)))"

" $\wedge$ dests sip. msg\_fresh  $\sigma$  (Rerr dests sip) =

( $\forall$  ripc  $\in$  dom(dests). (ripic  $\in$  kD(rt ( $\sigma$  sip))

$\wedge$  the (dests ripc) - 1  $\leq$  nsqn (rt ( $\sigma$  sip)) ripc))"

" $\wedge$ d dip. msg\_fresh  $\sigma$  (Newpkt d dip) = True"

" $\wedge$ dip sip. msg\_fresh  $\sigma$  (Pkt d dip sip) = True"

unfolding msg\_fresh\_def by simp\_all

lemma msg\_fresh\_inc\_sn [simp, elim]:

"msg\_fresh  $\sigma$  m  $\Longrightarrow$  rreq\_rrep\_sn m"

by (cases m) simp\_all

lemma recv\_msg\_fresh\_inc\_sn [simp, elim]:

"orecvmsg (msg\_fresh)  $\sigma$  m  $\Longrightarrow$  recvmmsg rreq\_rrep\_sn m"

by (cases m) simp\_all

lemma rreq\_nsqn\_is\_fresh [simp]:

fixes  $\sigma$  msg hops rreqid dip dsn dsk oip osn sip handled

assumes "rreq\_rrep\_fresh (rt ( $\sigma$  sip)) (Rreq hops rreqid dip dsn dsk oip osn sip handled)"

and "rreq\_rrep\_sn (Rreq hops rreqid dip dsn dsk oip osn sip handled)"

shows "msg\_fresh  $\sigma$  (Rreq hops rreqid dip dsn dsk oip osn sip handled)"

(is "msg\_fresh  $\sigma$  ?msg")

proof -

let ?rt = "rt ( $\sigma$  sip)"

from assms(2) have "1  $\leq$  osn" by simp

thus ?thesis

unfolding msg\_fresh\_def

```

proof (simp only: msg.case, intro conjI impI)
  assume "sip ≠ oip"
  with assms(1) show "oip ∈ kD(?rt)" by simp
next
  assume "sip ≠ oip"
  and "nsqn ?rt oip = osn"
  show "the (dhops ?rt oip) ≤ hops ∨ the (flag ?rt oip) = inv"
  proof (cases "oip ∈ vD(?rt)")
    assume "oip ∈ vD(?rt)"
    hence "nsqn ?rt oip = sqn ?rt oip" ..
    with ⟨nsqn ?rt oip = osn⟩ have "sqn ?rt oip = osn" by simp
    with assms(1) and ⟨sip ≠ oip⟩ have "the (dhops ?rt oip) ≤ hops"
      by simp
    thus ?thesis ..
  next
    assume "oip ∉ vD(?rt)"
    moreover from assms(1) and ⟨sip ≠ oip⟩ have "oip ∈ kD(?rt)" by simp
    ultimately have "oip ∈ iD(?rt)" by auto
    hence "the (flag ?rt oip) = inv" ..
    thus ?thesis ..
  qed
next
  assume "sip ≠ oip"
  with assms(1) have "osn ≤ sqn ?rt oip" by auto
  thus "osn ≤ nsqn (rt (σ sip)) oip"
  proof (rule nat_le_eq_or_lt)
    assume "osn < sqn ?rt oip"
    hence "osn ≤ sqn ?rt oip - 1" by simp
    also have "... ≤ nsqn ?rt oip" by (rule sqn_nsqn)
    finally show "osn ≤ nsqn ?rt oip" .
  next
    assume "osn = sqn ?rt oip"
    with assms(1) and ⟨sip ≠ oip⟩ have "oip ∈ kD(?rt)"
      and "the (flag ?rt oip) = val"
      by auto
    hence "nsqn ?rt oip = sqn ?rt oip" ..
    with ⟨osn = sqn ?rt oip⟩ have "nsqn ?rt oip = osn" by simp
    thus "osn ≤ nsqn ?rt oip" by simp
  qed
qed simp
qed

lemma rrep_nsqn_is_fresh [simp]:
  fixes σ msg hops dip dsn oip sip
  assumes "rreq_rrep_fresh (rt (σ sip)) (Rrep hops dip dsn oip sip)"
  and "rreq_rrep_sn (Rrep hops dip dsn oip sip)"
  shows "msg_fresh σ (Rrep hops dip dsn oip sip)"
  (is "msg_fresh σ ?msg")
proof -
  let ?rt = "rt (σ sip)"
  from assms have "sip ≠ dip → dip ∈ kD(?rt) ∧ sqn ?rt dip = dsn ∧ the (flag ?rt dip) = val"
  by simp
  hence "sip ≠ dip → dip ∈ kD(?rt) ∧ nsqn ?rt dip ≥ dsn"
  by clarsimp
  with assms show "msg_fresh σ ?msg"
  by clarsimp
qed

lemma rerr_nsqn_is_fresh [simp]:
  fixes σ msg dests sip
  assumes "rerr_invalid (rt (σ sip)) (Rerr dests sip)"
  shows "msg_fresh σ (Rerr dests sip)"
  (is "msg_fresh σ ?msg")
proof -

```

```

let ?rt = "rt (σ sip)"
from assms have *: "(∀rip∈dom(dests). (rip∈iD(rt (σ sip))
      ∧ the (dests rip) = sqn (rt (σ sip)) rip))"
  by clarsimp
have "(∀rip∈dom(dests). (rip∈kD(rt (σ sip))
      ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip))"
proof
  fix rip
  assume "rip ∈ dom dests"
  with * have "rip∈iD(rt (σ sip))" and "the (dests rip) = sqn (rt (σ sip)) rip"
    by auto

  from this(2) have "the (dests rip) - 1 = sqn (rt (σ sip)) rip - 1" by simp
  also have "... ≤ nsqn (rt (σ sip)) rip" by (rule sqn_nsqn)
  finally have "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip" .

  with ⟨rip∈iD(rt (σ sip))⟩
    show "rip∈kD(rt (σ sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
      by clarsimp
qed
thus "msg_fresh σ ?msg"
  by simp
qed

lemma quality_increases_msg_fresh [elim]:
  assumes qinc: "∀j. quality_increases (σ j) (σ' j)"
    and "msg_fresh σ m"
  shows "msg_fresh σ' m"
using assms(2)
proof (cases m)
  fix hops rreqid dip dsn dsk oip osn sip handled
  assume [simp]: "m = Rreq hops rreqid dip dsn dsk oip osn sip handled"
    and "msg_fresh σ m"
  then have "osn ≥ 1" and "sip = oip ∨ (oip∈kD(rt (σ sip)) ∧ osn ≤ nsqn (rt (σ sip)) oip
      ∧ (nsqn (rt (σ sip)) oip = osn
      → (the (dhops (rt (σ sip)) oip) ≤ hops
      ∨ the (flag (rt (σ sip)) oip) = inv)))"

  by auto
from this(2) show ?thesis
proof
  assume "sip = oip" with ⟨osn ≥ 1⟩ show ?thesis by simp
next
  assume "oip∈kD(rt (σ sip)) ∧ osn ≤ nsqn (rt (σ sip)) oip
      ∧ (nsqn (rt (σ sip)) oip = osn
      → (the (dhops (rt (σ sip)) oip) ≤ hops
      ∨ the (flag (rt (σ sip)) oip) = inv))"
  moreover from qinc have "quality_increases (σ sip) (σ' sip)" ..
  ultimately have "oip∈kD(rt (σ' sip)) ∧ osn ≤ nsqn (rt (σ' sip)) oip
      ∧ (nsqn (rt (σ' sip)) oip = osn
      → (the (dhops (rt (σ' sip)) oip) ≤ hops
      ∨ the (flag (rt (σ' sip)) oip) = inv))"

  using ⟨osn ≥ 1⟩ by (rule quality_increases_rreq_rrep_props [rotated 2])
  with ⟨osn ≥ 1⟩ show "msg_fresh σ' m"
    by (clarsimp)
qed
next
  fix hops dip dsn oip sip
  assume [simp]: "m = Rrep hops dip dsn oip sip"
    and "msg_fresh σ m"
  then have "dsn ≥ 1" and "sip = dip ∨ (dip∈kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip
      ∧ (nsqn (rt (σ sip)) dip = dsn
      → (the (dhops (rt (σ sip)) dip) ≤ hops
      ∨ the (flag (rt (σ sip)) dip) = inv)))"

  by auto

```

```

from this(2) show "?thesis"
proof
  assume "sip = dip" with ⟨dsn ≥ 1⟩ show ?thesis by simp
next
  assume "dip ∈ kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip
        ∧ (nsqn (rt (σ sip)) dip = dsn
          → (the (dhops (rt (σ sip)) dip) ≤ hops
             ∨ the (flag (rt (σ sip)) dip) = inv))"
  moreover from qinc have "quality_increases (σ sip) (σ' sip)" ..
  ultimately have "dip ∈ kD(rt (σ' sip)) ∧ dsn ≤ nsqn (rt (σ' sip)) dip
                  ∧ (nsqn (rt (σ' sip)) dip = dsn
                    → (the (dhops (rt (σ' sip)) dip) ≤ hops
                       ∨ the (flag (rt (σ' sip)) dip) = inv))"

  using ⟨dsn ≥ 1⟩ by (rule quality_increases_rreq_rrep_props [rotated 2])
  with ⟨dsn ≥ 1⟩ show "msg_fresh σ' m"
  by clarsimp
qed
next
  fix dests sip
  assume [simp]: "m = Rerr dests sip"
  and "msg_fresh σ m"
  then have *: "∀ rip ∈ dom(dests). rip ∈ kD(rt (σ sip))
              ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"

  by simp
  have "∀ rip ∈ dom(dests). rip ∈ kD(rt (σ' sip))
      ∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip"

  proof
    fix rip
    assume "rip ∈ dom(dests)"
    with * have "rip ∈ kD(rt (σ sip))" and "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
      by - (drule(1) bspec, clarsimp)+
    moreover from qinc have "quality_increases (σ sip) (σ' sip)" by simp
    ultimately show "rip ∈ kD(rt (σ' sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip" ..
  qed
  thus ?thesis by simp
qed simp_all
end

```

## 4.8 The ‘open’ AODV model

```

theory D_OAodv
imports D_Aodv "../.../AWN/OAWN_SOS_Labels" "../.../AWN/OAWN_Convert"
begin

```

Definitions for stating and proving global network properties over individual processes.

```

definition  $\sigma_{AODV}'$  :: " $((ip \Rightarrow state) \times ((state, msg, pseqp, pseqp label) seqp)) set$ "
where " $\sigma_{AODV}' \equiv \{(\lambda i. aodv\_init\ i, \Gamma_{AODV}\ PAodv)\}$ "

```

**abbreviation** opaodv

```

:: " $ip \Rightarrow ((ip \Rightarrow state) \times (state, msg, pseqp, pseqp label) seqp, msg\ seq\_action)\ automaton$ "
where
"opaodv i  $\equiv$  ( $\lfloor$  init =  $\sigma_{AODV}'$ , trans = oseqp_sos  $\Gamma_{AODV}$  i  $\rfloor$ )"

```

```

lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))"
unfolding  $\sigma_{AODV}'\_def$   $\sigma_{AODV}'\_def$  by rule simp_all

```

```

lemma oaodv_control_within [simp]: "control_within  $\Gamma_{AODV}$  (init (opaodv i))"
unfolding  $\sigma_{AODV}'\_def$  by (rule control_withinI) (auto simp del:  $\Gamma_{AODV}'\_simps$ )

```

```

lemma  $\sigma_{AODV}'\_labels$  [simp]: " $(\sigma, p) \in \sigma_{AODV}' \implies labels\ \Gamma_{AODV}\ p = \{PAodv-:0\}$ "
unfolding  $\sigma_{AODV}'\_def$  by simp

```

```

lemma oaodv_init_kD_empty [simp]:

```

```
"( $\sigma$ ,  $p$ )  $\in$   $\sigma_{AODV}' \implies kD$  ( $rt$  ( $\sigma$   $i$ )) = {}"
unfolding  $\sigma_{AODV}'\_def$   $kD\_def$  by simp
```

```
lemma oaodv_init_vD_empty [simp]:
  " $(\sigma, p) \in \sigma_{AODV}' \implies vD$  ( $rt$  ( $\sigma$   $i$ )) = {}"
  unfolding  $\sigma_{AODV}'\_def$   $vD\_def$  by simp
```

```
lemma oaodv_trans: " $trans$  (opaodv  $i$ ) = oseqp_sos  $\Gamma_{AODV}$   $i$ "
  by simp
```

declare

```
oseq_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]
oseq_step_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]
```

end

## 4.9 Global invariant proofs over sequential processes

theory *D\_Global\_Invariants*

imports *D\_Seq\_Invariants*

*D\_Aodv\_Predicates*

*D\_Fresher*

*D\_Quality\_Increases*

"../../../../AWN/DAWN\_Convert"

*D\_OAodv*

begin

```
lemma other_quality_increases [elim]:
  assumes "other quality_increases  $I$   $\sigma$   $\sigma'$ "
  shows " $\forall j$ . quality_increases ( $\sigma$   $j$ ) ( $\sigma'$   $j$ )"
  using assms by (rule, clarsimp) (metis quality_increases_refl)
```

```
lemma weaken_otherwith [elim]:
  fixes  $m$ 
  assumes *: "otherwith  $P$   $I$  (orecvmsg  $Q$ )  $\sigma$   $\sigma'$   $a$ "
  and weakenP: " $\bigwedge \sigma m$ .  $P$   $\sigma$   $m \implies P'$   $\sigma$   $m$ "
  and weakenQ: " $\bigwedge \sigma m$ .  $Q$   $\sigma$   $m \implies Q'$   $\sigma$   $m$ "
  shows "otherwith  $P'$   $I$  (orecvmsg  $Q'$ )  $\sigma$   $\sigma'$   $a$ "
  proof
    fix  $j$ 
    assume " $j \notin I$ "
    with * have " $P$  ( $\sigma$   $j$ ) ( $\sigma'$   $j$ )" by auto
    thus " $P'$  ( $\sigma$   $j$ ) ( $\sigma'$   $j$ )" by (rule weakenP)
  next
    from * have "orecvmsg  $Q$   $\sigma$   $a$ " by auto
    thus "orecvmsg  $Q'$   $\sigma$   $a$ "
      by rule (erule weakenQ)
  qed
```

```
lemma oreceived_msg_inv:
  assumes other: " $\bigwedge \sigma \sigma' m$ . [ $P$   $\sigma$   $m$ ; other  $Q$   $\{i\}$   $\sigma$   $\sigma'$ ]  $\implies P$   $\sigma'$   $m$ "
  and local: " $\bigwedge \sigma m$ .  $P$   $\sigma$   $m \implies P$  ( $\sigma(i := \sigma i(\text{msg} := m))$ )  $m$ "
  shows "opaodv  $i \models (\text{otherwith } Q \{i\} (\text{orecvmsg } P), \text{other } Q \{i\} \rightarrow)$ 
  onl  $\Gamma_{AODV} (\lambda(\sigma, l). l \in \{PAodv-:1\} \rightarrow P \sigma (\text{msg} (\sigma i)))$ "
  proof (inv_cterms, intro impI)
    fix  $\sigma \sigma' l$ 
    assume " $l = PAodv-:1 \rightarrow P \sigma (\text{msg} (\sigma i))$ "
    and " $l = PAodv-:1$ "
    and "other  $Q \{i\}$   $\sigma$   $\sigma'$ "
    from this(1-2) have " $P \sigma (\text{msg} (\sigma i))$ " ..
    hence " $P \sigma' (\text{msg} (\sigma i))$ " using  $\langle \text{other } Q \{i\} \sigma \sigma' \rangle$ 
      by (rule other)
    moreover from  $\langle \text{other } Q \{i\} \sigma \sigma' \rangle$  have " $\sigma' i = \sigma i$ " ..
    ultimately show " $P \sigma' (\text{msg} (\sigma' i))$ " by simp
```



```

next
  fix  $\sigma \sigma' msg$ 
  assume "otherwith  $Q \{i\}$  (orecvmsg  $P$ )  $\sigma \sigma'$  (receive  $msg$ )"
  and " $\sigma' i = \sigma i(msg := msg)$ "
  from this(1) have " $P \sigma msg$ "
  and " $\forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j)$ " by auto
  from this(1) have " $P (\sigma(i := \sigma i(msg := msg))) msg$ " by (rule local)
  thus " $P \sigma' msg$ "
  proof (rule other)
    from  $\langle \sigma' i = \sigma i(msg := msg) \rangle$  and  $\langle \forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j) \rangle$ 
    show "other  $Q \{i\} (\sigma(i := \sigma i(msg := msg))) \sigma'$ "
    by - (rule otherI, auto)
  qed
qed

```

(Equivalent to) Proposition 7.27

lemma local\_quality\_increases:

```

"paadv  $i \models_A (recvmsg rreq_rrep\_sn \rightarrow) onll \Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). quality\_increases \xi \xi')$ "
proof (rule step_invariantI)
  fix  $s a s'$ 
  assume  $sr: "s \in reachable (paadv i) (recvmsg rreq_rrep\_sn)"$ 
  and  $tr: "(s, a, s') \in trans (paadv i)"$ 
  and  $rm: "recvmsg rreq_rrep\_sn a"$ 
  from  $sr$  have  $srTT: "s \in reachable (paadv i) TT" ..$ 

  from route_tables_fresher  $sr tr rm$ 
  have " $onll \Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \forall dip \in kD (rt \xi). rt \xi \sqsubseteq_{dip} rt \xi') (s, a, s')$ "
  by (rule step_invariantD)

  moreover from known_destinations_increase  $srTT tr TT\_True$ 
  have " $onll \Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). kD (rt \xi) \subseteq kD (rt \xi')) (s, a, s')$ "
  by (rule step_invariantD)

  moreover from sqns_increase  $srTT tr TT\_True$ 
  have " $onll \Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \forall ip. sqn (rt \xi) ip \leq sqn (rt \xi') ip) (s, a, s')$ "
  by (rule step_invariantD)

  ultimately show " $onll \Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). quality\_increases \xi \xi') (s, a, s')$ "
  unfolding onll_def by auto
qed

```

lemmas olocal\_quality\_increases =

```

open_seq_step_invariant [OF local_quality_increases initiali_aadv oaadv_trans aadv_trans,
  simplified seql_onll_swap]

```

lemma oquality\_increases:

```

"opaadv  $i \models_A (otherwith quality\_increases \{i\} (orecvmsg (\lambda_. rreq\_rrep\_sn)),$ 
  other quality\_increases  $\{i\} \rightarrow)$ 
  onll  $\Gamma_{AODV} (\lambda((\sigma, \_), \_, (\sigma', \_)). \forall j. quality\_increases (\sigma j) (\sigma' j))"$ 
(is " $\_ \models_A (?S, \_ \rightarrow) \_$ ")
proof (rule onll_ostep_invariantI, simp)
  fix  $\sigma p l a \sigma' p' l'$ 
  assume  $or: "(\sigma, p) \in oreachable (opaadv i) ?S (other quality\_increases \{i\})"$ 
  and  $ll: "l \in labels \Gamma_{AODV} p"$ 
  and " $?S \sigma \sigma' a$ "
  and  $tr: "((\sigma, p), a, (\sigma', p')) \in oseqp\_sos \Gamma_{AODV} i"$ 
  and  $ll': "l' \in labels \Gamma_{AODV} p'"$ 
  from this(1-3) have " $orecvmsg (\lambda_. rreq\_rrep\_sn) \sigma a$ "
  by (auto dest!: oreachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"
    and QU="other quality_increases {i}"]
    otherwith_actionD)
  with  $or$  have  $orw: "(\sigma, p) \in oreachable (opaadv i) (act (recvmsg rreq_rrep_sn))$ 
    (other quality_increases  $\{i\})"$ 
  by - (erule oreachable_weakenE, auto)

```

with tr ll ll' and (orecvmsg ( $\lambda\_.$  rreq\_rrep\_sn)  $\sigma$  a) have "quality\_increases ( $\sigma$  i) ( $\sigma'$  i)"  
 by - (drule onll\_ostep\_invariantD [OF olocal\_quality\_increases], auto simp: seqll\_def)  
 with (?S  $\sigma$   $\sigma'$  a) show " $\forall j.$  quality\_increases ( $\sigma$  j) ( $\sigma'$  j)"  
 by (auto dest!: otherwith\_syncD)

qed

lemma rreq\_rrep\_nsqn\_fresh\_any\_step\_invariant:

"opaadv i  $\models_A$  (act (recvmsg rreq\_rrep\_sn), other A {i}  $\rightarrow$ )  
 onll  $\Gamma_{AODV}$  ( $\lambda((\sigma, \_), a, \_).$  anycast (msg\_fresh  $\sigma$ ) a)"

proof (rule ostep\_invariantI, simp del: act\_simp)

fix  $\sigma$  p a  $\sigma'$  p'

assume or: " $(\sigma, p) \in$  oreachable (opaadv i) (act (recvmsg rreq\_rrep\_sn)) (other A {i})"  
 and " $((\sigma, p), a, (\sigma', p')) \in$  oseqp\_sos  $\Gamma_{AODV}$  i"  
 and recv: "act (recvmsg rreq\_rrep\_sn)  $\sigma$   $\sigma'$  a"

obtain l l' where " $l \in$ labels  $\Gamma_{AODV}$  p" and " $l' \in$ labels  $\Gamma_{AODV}$  p'"  
 by (metis aadv\_ex\_label)

from  $((\sigma, p), a, (\sigma', p')) \in$  oseqp\_sos  $\Gamma_{AODV}$  i  
 have tr: " $((\sigma, p), a, (\sigma', p')) \in$  trans (opaadv i)" by simp

have "anycast (rreq\_rrep\_fresh (rt ( $\sigma$  i))) a"

proof -

have "opaadv i  $\models_A$  (act (recvmsg rreq\_rrep\_sn), other A {i}  $\rightarrow$ )  
 onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_).$  anycast (rreq\_rrep\_fresh (rt  $\xi$ )) a))"

by (rule ostep\_invariant\_weakenE [OF  
 open\_seq\_step\_invariant [OF rreq\_rrep\_fresh\_any\_step\_invariant initiali\_aadv,  
 simplified seqll\_onll\_swap]]) auto

hence "onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_).$  anycast (rreq\_rrep\_fresh (rt  $\xi$ )) a))  
 ( $(\sigma, p), a, (\sigma', p')$ )"

using or tr recv by - (erule(4) ostep\_invariantE)

thus ?thesis

using  $\langle l \in$ labels  $\Gamma_{AODV}$  p  $\rangle$  and  $\langle l' \in$ labels  $\Gamma_{AODV}$  p'  $\rangle$  by auto

qed

moreover have "anycast (rerr\_invalid (rt ( $\sigma$  i))) a"

proof -

have "opaadv i  $\models_A$  (act (recvmsg rreq\_rrep\_sn), other A {i}  $\rightarrow$ )  
 onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_).$  anycast (rerr\_invalid (rt  $\xi$ )) a))"

by (rule ostep\_invariant\_weakenE [OF  
 open\_seq\_step\_invariant [OF rerr\_invalid\_any\_step\_invariant initiali\_aadv,  
 simplified seqll\_onll\_swap]]) auto

hence "onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_).$  anycast (rerr\_invalid (rt  $\xi$ )) a))  
 ( $(\sigma, p), a, (\sigma', p')$ )"

using or tr recv by - (erule(4) ostep\_invariantE)

thus ?thesis

using  $\langle l \in$ labels  $\Gamma_{AODV}$  p  $\rangle$  and  $\langle l' \in$ labels  $\Gamma_{AODV}$  p'  $\rangle$  by auto

qed

moreover have "anycast rreq\_rrep\_sn a"

proof -

from or tr recv

have "onll  $\Gamma_{AODV}$  (seqll i ( $\lambda(\_, a, \_).$  anycast rreq\_rrep\_sn a)) ( $(\sigma, p), a, (\sigma', p')$ )"  
 by (rule ostep\_invariantE [OF

open\_seq\_step\_invariant [OF rreq\_rrep\_sn\_any\_step\_invariant initiali\_aadv  
 oaadv\_trans aadv\_trans,  
 simplified seqll\_onll\_swap]])

thus ?thesis

using  $\langle l \in$ labels  $\Gamma_{AODV}$  p  $\rangle$  and  $\langle l' \in$ labels  $\Gamma_{AODV}$  p'  $\rangle$  by auto

qed

moreover have "anycast ( $\lambda m.$  not\_Pkt m  $\rightarrow$  msg\_sender m = i) a"

proof -

have "opaadv i  $\models_A$  (act (recvmsg rreq\_rrep\_sn), other A {i}  $\rightarrow$ )  
 onll  $\Gamma_{AODV}$  (seqll i ( $\lambda((\xi, \_), a, \_).$  anycast ( $\lambda m.$  not\_Pkt m  $\rightarrow$  msg\_sender m = i) a))"  
 by (rule ostep\_invariant\_weakenE [OF

```

      open_seq_step_invariant [OF sender_ip_valid initiali_aodv,
                              simplified seqll_onll_swap]]) auto
    thus ?thesis using or tr recv ⟨l∈labels ΓAODV p⟩ and ⟨l'∈labels ΓAODV p'⟩
      by - (drule(3) onll_ostep_invariantD, auto)
qed

ultimately have "anycast (msg_fresh σ) a"
  by (simp_all add: anycast_def
      del: msg_fresh
      split: seq_action.split_asm msg.split_asm) simp_all
thus "onll ΓAODV (λ((σ, _), a, _). anycast (msg_fresh σ) a) ((σ, p), a, (σ', p'))"
  by auto
qed

lemma oreceived_rreq_rrep_nsqn_fresh_inv:
  "opaodv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
              other quality_increases {i} →)
              onl ΓAODV (λ(σ, l). l ∈ {PAodv-:1} → msg_fresh σ (msg (σ i)))"
proof (rule oreceived_msg_inv)
  fix σ σ' m
  assume *: "msg_fresh σ m"
  and "other quality_increases {i} σ σ'"
  from this(2) have "∀j. quality_increases (σ j) (σ' j)" ..
  thus "msg_fresh σ' m" using * ..
next
  fix σ m
  assume "msg_fresh σ m"
  thus "msg_fresh (σ(i := σ i (msg := m))) m"
proof (cases m)
  fix dests sip
  assume "m = Rerr dests sip"
  with ⟨msg_fresh σ m⟩ show ?thesis by auto
qed auto
qed

lemma oquality_increases_nsqn_fresh:
  "opaodv i ⊨A (otherwith quality_increases {i} (orecvmsg msg_fresh),
              other quality_increases {i} →)
              onll ΓAODV (λ((σ, _), _, (σ', _)). ∀j. quality_increases (σ j) (σ' j))"
  by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
  "opaodv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
              other quality_increases {i} →)
              onl ΓAODV (seq1 i (λ(ξ, l). l ∈ {PAodv-:4, PAodv-:5} ∪ {PRreq-:n | n. True} → 1 ≤ osn ξ))"
  by (rule oinvariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_aodv]])
  (auto simp: seq1_onll_swap)

lemma rreq_sip:
  "opaodv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
              other quality_increases {i} →)
              onl ΓAODV (λ(σ, l).
                (l ∈ {PAodv-:4, PAodv-:5, PRreq-:0, PRreq-:2} ∧ sip (σ i) ≠ oip (σ i))
                → oip (σ i) ∈ kD(rt (σ (sip (σ i))))
                  ∧ nsqn (rt (σ (sip (σ i)))) (oip (σ i)) ≥ osn (σ i)
                  ∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ i)) = osn (σ i)
                    → (hops (σ i) ≥ the (dhops (rt (σ (sip (σ i)))) (oip (σ i)))
                      ∨ the (flag (rt (σ (sip (σ i)))) (oip (σ i))) = inv)))"
  (is "_ ⊨ (?S, ?U →) _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh
                                                       aodv_wf oaodv_trans]
      onl_oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
      onl_oinvariant_sterms [OF aodv_wf oosn_rreq]
      simp add: seqlsimp)

```

```

      simp del: One_nat_def, rule impI)
fix  $\sigma \sigma' p l$ 
assume "( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U"
  and "l  $\in$  labels  $\Gamma_{AODV} p$ "
  and pre:
    "(l = PAadv-:4  $\vee$  l = PAadv-:5  $\vee$  l = PRreq-:0  $\vee$  l = PRreq-:2)  $\wedge$  sip ( $\sigma$  i)  $\neq$  oip ( $\sigma$  i)
     $\rightarrow$  oip ( $\sigma$  i)  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
     $\wedge$  osn ( $\sigma$  i)  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma$  i))
     $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma$  i)) = osn ( $\sigma$  i)
     $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma$  i)))  $\leq$  hops ( $\sigma$  i)
     $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma$  i))) = inv)"
  and "other quality_increases {i}  $\sigma \sigma'$ "
  and hyp: "(l=PAadv-:4  $\vee$  l=PAadv-:5  $\vee$  l=PRreq-:0  $\vee$  l=PRreq-:2)  $\wedge$  sip ( $\sigma'$  i)  $\neq$  oip ( $\sigma'$  i)"
    (is "?labels  $\wedge$  sip ( $\sigma'$  i)  $\neq$  oip ( $\sigma'$  i)")
from this(4) have " $\sigma' i = \sigma i$ " ..
with hyp have hyp': "?labels  $\wedge$  sip ( $\sigma$  i)  $\neq$  oip ( $\sigma$  i)" by simp
show "oip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma'$  (sip ( $\sigma' i$ ))))
 $\wedge$  osn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (oip ( $\sigma' i$ ))
 $\wedge$  (nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (oip ( $\sigma' i$ )) = osn ( $\sigma' i$ )
 $\rightarrow$  the (dhops (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (oip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
 $\vee$  the (flag (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (oip ( $\sigma' i$ ))) = inv)"
proof (cases "sip ( $\sigma$  i) = i")
  assume "sip ( $\sigma$  i)  $\neq$  i"
  from (other quality_increases {i}  $\sigma \sigma'$ )
    have "quality_increases ( $\sigma$  (sip ( $\sigma$  i))) ( $\sigma'$  (sip ( $\sigma' i$ )))"
      by (rule otherE) (clarsimp simp: (sip ( $\sigma$  i)  $\neq$  i))
  moreover from (( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U) (l  $\in$  labels  $\Gamma_{AODV} p$ ) and hyp
    have "l  $\leq$  osn ( $\sigma' i$ )"
      by (auto dest!: onl_oainvariant_weakenD [OF oosn_rreq]
        simp add: seqlsimp ( $\sigma' i = \sigma i$ ))
  moreover from (sip ( $\sigma$  i)  $\neq$  i) hyp' and pre
    have "oip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
     $\wedge$  osn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma' i$ ))
     $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma' i$ )) = osn ( $\sigma' i$ )
     $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
     $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma' i$ ))) = inv)"
      by (auto simp: ( $\sigma' i = \sigma i$ ))
  ultimately show ?thesis
    by (rule quality_increases_rreq_rrep_props)
next
  assume "sip ( $\sigma$  i) = i" thus ?thesis
    using ( $\sigma' i = \sigma i$ ) hyp and pre by auto
qed
qed (auto elim!: quality_increases_rreq_rrep_props')

```

```

lemma odsn_rrep:
  "opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i}  $\rightarrow$ )
    onl  $\Gamma_{AODV}$  (seq1 i ( $\lambda(\xi, l). l \in \{PAadv-:6, PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow l \leq dsn \xi$ ))"
  by (rule oinvariant_weakenE [OF open_seq_invariant [OF dsn_rrep initiali_aadv]])
  (auto simp: seq1_onl_swap)

```

```

lemma rrep_sip:
  "opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i}  $\rightarrow$ )
    onl  $\Gamma_{AODV}$  ( $\lambda(\sigma, l).$ 
      (l  $\in$  {PAadv-:6, PAadv-:7, PRrep-:0, PRrep-:1}  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i))
       $\rightarrow$  dip ( $\sigma$  i)  $\in$  kD(rt ( $\sigma$  (sip ( $\sigma$  i))))
       $\wedge$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))  $\geq$  dsn ( $\sigma$  i)
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i)) = dsn ( $\sigma$  i)
       $\rightarrow$  (hops ( $\sigma$  i)  $\geq$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))))
       $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = inv))"
  (is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf

```

```

                                oaadv_trans]
                                onl_oInvariant_sterms [OF aadv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
                                onl_oInvariant_sterms [OF aadv_wf odsn_rrep]
                                simp del: One_nat_def, rule impI)
fix  $\sigma \sigma' p l$ 
assume "( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U"
  and "l  $\in$  labels  $\Gamma_{AODV} p$ "
  and pre:
    "(l = PAadv-:6  $\vee$  l = PAadv-:7  $\vee$  l = PRrep-:0  $\vee$  l = PRrep-:1)  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i)
     $\rightarrow$  dip ( $\sigma$  i)  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
       $\wedge$  dsn ( $\sigma$  i)  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i)) = dsn ( $\sigma$  i)
         $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i)))  $\leq$  hops ( $\sigma$  i)
           $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = inv)"
  and "other quality_increases {i}  $\sigma \sigma'$ "
  and hyp: "(l=PAadv-:6  $\vee$  l=PAadv-:7  $\vee$  l=PRrep-:0  $\vee$  l=PRrep-:1)  $\wedge$  sip ( $\sigma'$  i)  $\neq$  dip ( $\sigma'$  i)"
    (is "?labels  $\wedge$  sip ( $\sigma'$  i)  $\neq$  dip ( $\sigma'$  i)")
from this(4) have " $\sigma' i = \sigma i$ " ..
with hyp have hyp': "?labels  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i)" by simp
show "dip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma'$  (sip ( $\sigma' i$ ))))
   $\wedge$  dsn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ ))
   $\wedge$  (nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ )) = dsn ( $\sigma' i$ ))
   $\rightarrow$  the (dhops (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
     $\vee$  the (flag (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ ))) = inv)"
proof (cases "sip ( $\sigma$  i) = i")
  assume "sip ( $\sigma$  i)  $\neq$  i"
  from (other quality_increases {i}  $\sigma \sigma'$ )
    have "quality_increases ( $\sigma$  (sip ( $\sigma$  i))) ( $\sigma'$  (sip ( $\sigma' i$ )))"
      by (rule otherE) (clarsimp simp: (sip ( $\sigma$  i)  $\neq$  i))
  moreover from (( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U) (l  $\in$  labels  $\Gamma_{AODV} p$ ) and hyp
    have "l  $\leq$  dsn ( $\sigma' i$ )"
      by (auto dest!: onl_oInvariant_weakenD [OF odsn_rrep]
          simp add: seqsimp (sip ( $\sigma$  i)  $\neq$  i))
  moreover from (sip ( $\sigma$  i)  $\neq$  i) hyp' and pre
    have "dip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
       $\wedge$  dsn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ ))
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ )) = dsn ( $\sigma' i$ ))
       $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
         $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ ))) = inv)"
    by (auto simp: (sip ( $\sigma$  i)  $\neq$  i))
  ultimately show ?thesis
    by (rule quality_increases_rreq_rrep_props)
next
  assume "sip ( $\sigma$  i) = i" thus ?thesis
    using (sip ( $\sigma$  i) = i) hyp and pre by auto
qed
qed (auto simp add: seqsimp elim!: quality_increases_rreq_rrep_props')

```

lemma rerr\_sip:

```

"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onl  $\Gamma_{AODV} (\lambda(\sigma, l).$ 
    l  $\in$  {PAadv-:8, PAadv-:9, PRerr-:0, PRerr-:1}
     $\rightarrow$  ( $\forall ripc \in dom(dests (\sigma i)). ripc \in kD(rt (\sigma (sip (\sigma i)))) \wedge$ 
      the (dests ( $\sigma$  i) ripc) - 1  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) ripc))"
(is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")

```

proof -

```

{ fix dests rip sip rsn and  $\sigma \sigma' :: "ip \Rightarrow state"$ 
  assume qinc: " $\forall j. quality\_increases (\sigma j) (\sigma' j)"$ 
    and *: " $\forall rip \in dom dests. rip \in kD (rt (\sigma sip))$ 
       $\wedge$  the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma sip$ )) rip"
    and "dests rip = Some rsn"
  from this(3) have "rip  $\in$  dom dests" by auto
  with * and (dests rip = Some rsn) have "rip  $\in$  kD (rt ( $\sigma sip$ ))"

```

and "rsn - 1 ≤ nsqn (rt (σ sip)) rip"

```

by (auto dest!: bspec)
from qinc have "quality_increases (σ sip) (σ' sip)" ..
have "rip ∈ kD(rt (σ' sip)) ∧ rsn - 1 ≤ nsqn (rt (σ' sip)) rip"
proof
  from ⟨rip ∈ kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  show "rip ∈ kD(rt (σ' sip))" ..
next
  from ⟨rip ∈ kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  have "nsqn (rt (σ sip)) rip ≤ nsqn (rt (σ' sip)) rip" ..
  with ⟨rsn - 1 ≤ nsqn (rt (σ sip)) rip⟩ show "rsn - 1 ≤ nsqn (rt (σ' sip)) rip"
  by (rule le_trans)
qed
} note partial = this

show ?thesis
by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
  oadv_trans]
  onl_oinvariant_sterms [OF aadv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
  other_quality_increases_other_localD
  simp del: One_nat_def, intro conjI]
  (clarsimp simp del: One_nat_def split: split_if_asm option.split_asm, erule(2) partial)+
qed

```

```

lemma prerr_guard: "paadv i ⊨
  onl ΓAODV (λ(ξ, l). (l = PRerr-:1
    → (∀ ip ∈ dom(dests ξ). ip ∈ vD(rt ξ)
      ∧ the (nhop (rt ξ) ip) = sip ξ
      ∧ sqn (rt ξ) ip < the (dests ξ ip))))"
by (inv_cterms) (clarsimp split: option.split_asm split_if_asm)

```

```

lemmas oaddpreRT_welldefined =
  open_seq_invariant [OF addpreRT_welldefined initiali_aadv oaadv_trans aadv_trans,
    simplified seq_l_onl_swap,
    THEN oinvariant_anyact]

```

```

lemmas odests_vD_inc_sqn =
  open_seq_invariant [OF dests_vD_inc_sqn initiali_aadv oaadv_trans aadv_trans,
    simplified seq_l_onl_swap,
    THEN oinvariant_anyact]

```

```

lemmas oprerr_guard =
  open_seq_invariant [OF prerr_guard initiali_aadv oaadv_trans aadv_trans,
    simplified seq_l_onl_swap,
    THEN oinvariant_anyact]

```

Proposition 7.28

```

lemma seq_compare_next_hop':
  "opaadv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} → onl ΓAODV (λ(σ, _).
      ∀ dip. let nhop = the (nhop (rt (σ i)) dip)
        in dip ∈ kD(rt (σ i)) ∧ nhop ≠ dip →
          dip ∈ kD(rt (σ nhop)) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhop)) dip)"
(is "_ ⊨ (?S, ?U →) _")
proof -
{ fix nhop and σ σ' :: "ip ⇒ state"
  assume pre: "∀ dip ∈ kD(rt (σ i)). nhop dip ≠ dip →
    dip ∈ kD(rt (σ (nhop dip))) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
  and qinc: "∀ j. quality_increases (σ j) (σ' j)"
  have "∀ dip ∈ kD(rt (σ i)). nhop dip ≠ dip →
    dip ∈ kD(rt (σ' (nhop dip))) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
  proof (intro ballI impI)
    fix dip

```

```

assume "dip∈kD(rt (σ i))"
  and "nhop dip ≠ dip"
with pre have "dip∈kD(rt (σ (nhop dip)))"
  and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
  by auto
from qinc have qinc_nhop: "quality_increases (σ (nhop dip)) (σ' (nhop dip))" ..
with ⟨dip∈kD(rt (σ (nhop dip)))⟩ have "dip∈kD (rt (σ' (nhop dip)))" ..

moreover have "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
proof -
  from ⟨dip∈kD(rt (σ (nhop dip)))⟩ qinc_nhop
  have "nsqn (rt (σ (nhop dip))) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ..
  with ⟨nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip⟩ show ?thesis
  by simp
qed

ultimately show "dip∈kD(rt (σ' (nhop dip)))
  ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ..
qed
} note basic = this

{ fix nhop and σ σ' :: "ip ⇒ state"
  assume pre: "∀dip∈kD(rt (σ i)). nhop dip ≠ dip → dip∈kD(rt (σ (nhop dip)))
    ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
  and ndest: "∀ripc∈dom (dests (σ i)). ripc ∈ kD (rt (σ (sip (σ i))))
    ∧ the (dests (σ i) ripc) - 1 ≤ nsqn (rt (σ (sip (σ i)))) ripc"
  and issip: "∀ip∈dom (dests (σ i)). nhop ip = sip (σ i)"
  and qinc: "∀j. quality_increases (σ j) (σ' j)"
  have "∀dip∈kD(rt (σ i)). nhop dip ≠ dip → dip ∈ kD (rt (σ' (nhop dip)))
    ∧ nsqn (invalidate (rt (σ i)) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
  proof (intro ballI impI)
    fix dip
    assume "dip∈kD(rt (σ i))"
      and "nhop dip ≠ dip"
    with pre and qinc have "dip∈kD(rt (σ' (nhop dip)))"
      and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
    by (auto dest!: basic)

    have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
    proof (cases "dip∈dom (dests (σ i))")
      assume "dip∈dom (dests (σ i))"
      with ⟨dip∈kD(rt (σ i))⟩ obtain dsn where "dests (σ i) dip = Some dsn"
        by auto
      with ⟨dip∈kD(rt (σ i))⟩ have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = dsn - 1"
        by (rule nsqn_invalidate_eq)
      moreover have "dsn - 1 ≤ nsqn (rt (σ' (nhop dip))) dip"
    proof -
      from ⟨dests (σ i) dip = Some dsn⟩ have "the (dests (σ i) dip) = dsn" by simp
      with ndest and ⟨dip∈dom (dests (σ i))⟩ have "dip ∈ kD (rt (σ (sip (σ i))))"
        "dsn - 1 ≤ nsqn (rt (σ (sip (σ i)))) dip"
      by auto
      moreover from issip and ⟨dip∈dom (dests (σ i))⟩ have "nhop dip = sip (σ i)" ..
      ultimately have "dip ∈ kD (rt (σ (nhop dip)))"
        and "dsn - 1 ≤ nsqn (rt (σ (nhop dip))) dip" by auto
      with qinc show "dsn - 1 ≤ nsqn (rt (σ' (nhop dip))) dip"
        by simp (metis kD_nsqn_quality_increases_trans)
    qed
    ultimately show ?thesis by simp
  next
  assume "dip ∉ dom (dests (σ i))"
  with ⟨dip∈kD(rt (σ i))⟩
  have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = nsqn (rt (σ i)) dip"
    by (rule nsqn_invalidate_other)
  with ⟨nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip⟩ show ?thesis by simp
}

```

```

qed
with ⟨dip∈kD(rt (σ' (nhop dip)))⟩
  show "dip ∈ kD (rt (σ' (nhop dip)))
    ∧ nsqn (invalidate (rt (σ i)) (dests (σ i))) dip ≤ nsqn (rt (σ' (nhop dip))) dip" ..
qed
} note basic_prerr = this

{ fix σ σ' :: "ip ⇒ state"
  assume a1: "∀dip∈kD(rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip∈kD(rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
  and a2: "∀j. quality_increases (σ j) (σ' j)"
  have "∀dip∈kD(rt (σ i)).
    the (nhop (update (rt (σ i)) (sip (σ i)) (0, unk, val, Suc 0, sip (σ i), { }))) dip ≠ dip →
    dip∈kD(rt (σ' (the (nhop (update (rt (σ i)) (sip (σ i))
      (0, unk, val, Suc 0, sip (σ i), { })))
      dip)))) ∧
    nsqn (update (rt (σ i)) (sip (σ i)) (0, unk, val, Suc 0, sip (σ i), { }))) dip
    ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) (sip (σ i))
      (0, unk, val, Suc 0, sip (σ i), { })))
      dip))))
    dip" (is "∀dip∈kD(rt (σ i)). ?P dip")
proof
  fix dip
  assume "dip∈kD(rt (σ i))"
  with a1 and a2
    have "the (nhop (rt (σ i)) dip) ≠ dip → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
      by - (drule(1) basic, auto)
  thus "?P dip" by (cases "dip = sip (σ i)") auto
qed
} note nhop_update_sip = this

{ fix σ σ' oip sip osn hops
  assume pre: "∀dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip∈kD(rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
  and qinc: "∀j. quality_increases (σ j) (σ' j)"
  and *: "sip ≠ oip → oip∈kD(rt (σ sip))
    ∧ osn ≤ nsqn (rt (σ sip)) oip
    ∧ (nsqn (rt (σ sip)) oip = osn
      → the (dhops (rt (σ sip)) oip) ≤ hops
      ∨ the (flag (rt (σ sip)) oip) = inv)"
  from pre and qinc
    have pre': "∀dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
      → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
      by (rule basic)
  have "(the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }))) oip) ≠ oip
    → oip∈kD(rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip, { }))) oip))))
    ∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }))) oip
    ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip, { }))) oip)))) oip"
    (is "?nhop_not_oip → ?oip_in_kD ∧ ?nsqn_le_nsqn")
  proof (rule, split update_rt_split_asm)
    assume "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })"
    and "the (nhop (rt (σ i)) oip) ≠ oip"
    with pre' show "?oip_in_kD ∧ ?nsqn_le_nsqn" by auto
  next
    assume rnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })"
    and notoip: ?nhop_not_oip
    with * qinc have ?oip_in_kD
      by (clarsimp elim!: kD_quality_increases)

```



```

    moreover with * pre qinc rtnot notoiip have ?nsqn_le_nsqn
    by simp (metis kD_nsqn_quality_increases_trans)
    ultimately show "?oip_in_kD ∧ ?nsqn_le_nsqn" ..
  qed
} note update1 = this

{ fix σ σ' oip sip osn hops
  assume pre: "∀dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip∈kD(rt (σ (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"
  and qinc: "∀j. quality_increases (σ j) (σ' j)"
  and *: "sip ≠ oip → oip∈kD(rt (σ sip))
    ∧ osn ≤ nsqn (rt (σ sip)) oip
    ∧ (nsqn (rt (σ sip)) oip = osn
      → the (dhops (rt (σ sip)) oip) ≤ hops
        ∨ the (flag (rt (σ sip)) oip) = inv)"

  from pre and qinc
  have pre': "∀dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
    → dip∈kD(rt (σ' (the (nhop (rt (σ i)) dip))))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
  by (rule basic)
  have "∀dip∈kD(rt (σ i)).
    the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip
    → dip∈kD(rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip, {})) dip))))
      ∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip
      ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip
        (osn, kno, val, Suc hops, sip, {})) dip)))) dip"
    (is "∀dip∈kD(rt (σ i)). _ → ?dip_in_kD dip ∧ ?nsqn_le_nsqn dip")
  proof (intro ballI impI, split update_rt_split_asm)
  fix dip
  assume "dip∈kD(rt (σ i))"
  and "the (nhop (rt (σ i)) dip) ≠ dip"
  and "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"
  with pre' show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip" by simp
  next
  fix dip
  assume "dip∈kD(rt (σ i))"
  and notdip: "the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip"
  and rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"
  show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip"
  proof (cases "dip = oip")
  assume "dip ≠ oip"
  with pre' (dip∈kD(rt (σ i))) notdip
  show ?thesis by clarsimp
  next
  assume "dip = oip"
  with rtnot qinc (dip∈kD(rt (σ i))) notdip *
  have "?dip_in_kD dip"
  by simp (metis kD_quality_increases)
  moreover from (dip = oip) rtnot qinc (dip∈kD(rt (σ i))) notdip *
  have "?nsqn_le_nsqn dip" by simp (metis kD_nsqn_quality_increases_trans)
  ultimately show ?thesis ..
  qed
  qed
} note update2 = this

have "opaadv i ⊨ (?S, ?U →) onl ΓAODV (λ(σ, _).
  ∀dip ∈ kD(rt (σ i)). the (nhop (rt (σ i)) dip) ≠ dip
  → dip ∈ kD(rt (σ (the (nhop (rt (σ i)) dip))))
  ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip)"
by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
  oadv_trans])

```

```

onl_oinvariant_sterms [OF aadv_wf oaddpreRT_welldefined]
onl_oinvariant_sterms [OF aadv_wf odests_vD_inc_sqn]
onl_oinvariant_sterms [OF aadv_wf oprerr_guard]
onl_oinvariant_sterms [OF aadv_wf rreq_sip]
onl_oinvariant_sterms [OF aadv_wf rrep_sip]
onl_oinvariant_sterms [OF aadv_wf rerr_sip]
other_quality_increases
other_localD
solve: basic basic_prerr
simp add: seqlsimp nsqn_invalidate nhop_update_sip
simp del: One_nat_def
(rule conjI, erule(2) update1, erule(2) update2)+

```

thus ?thesis unfolding Let\_def by auto  
qed

Proposition 7.30

```

lemmas okD_unk_or_atleast_one =
  open_seq_invariant [OF kD_unk_or_atleast_one initiali_aadv,
    simplified seql_onl_swap]

```

```

lemmas ozero_seq_unk_hops_one =
  open_seq_invariant [OF zero_seq_unk_hops_one initiali_aadv,
    simplified seql_onl_swap]

```

lemma oreachable\_fresh\_okD\_unk\_or\_atleast\_one:

```

fixes dip
assumes "(σ, p) ∈ oreachable (opaadv i)
  (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
    ∧ msg_zhops m)))
  (other quality_increases {i})"
and "dip ∈ kD(rt (σ i))"
shows "π3(the (rt (σ i) dip)) = unk ∨ 1 ≤ π2(the (rt (σ i) dip))"
(is "?P dip")
proof -
have "∃ l. l ∈ labels ΓAADV p" by (metis aadv_ex_label)
with assms(1) have "∀ dip ∈ kD (rt (σ i)). ?P dip"
by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaadv_trans aadv_trans]],
  auto dest!: otherwith_actionD onID simp: seqlsimp)
with ⟨dip ∈ kD(rt (σ i))⟩ show ?thesis by simp
qed

```

lemma oreachable\_fresh\_ozero\_seq\_unk\_hops\_one:

```

fixes dip
assumes "(σ, p) ∈ oreachable (opaadv i)
  (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
    ∧ msg_zhops m)))
  (other quality_increases {i})"
and "dip ∈ kD(rt (σ i))"
shows "sqn (rt (σ i) dip) = 0 →
  sqnf (rt (σ i) dip) = unk
  ∧ the (dhops (rt (σ i) dip)) = 1
  ∧ the (nhop (rt (σ i) dip)) = dip"
(is "?P dip")
proof -
have "∃ l. l ∈ labels ΓAADV p" by (metis aadv_ex_label)
with assms(1) have "∀ dip ∈ kD (rt (σ i)). ?P dip"
by - (drule oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaadv_trans aadv_trans]],
  auto dest!: onID otherwith_actionD simp: seqlsimp)
with ⟨dip ∈ kD(rt (σ i))⟩ show ?thesis by simp
qed

```

lemma seq\_nhop\_quality\_increases':

```

shows "opaadv i ⊨ (otherwith (op=) {i})

```

```

(orecvmsg ( $\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m$ )),
  other quality_increases {i}  $\rightarrow$ )
onl  $\Gamma_{AODV} (\lambda(\sigma, \_). \forall \text{dip}. \text{let } \text{nhip} = \text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})$ 
  in  $\text{dip} \in \text{vD } (\text{rt } (\sigma \text{ i})) \cap \text{vD } (\text{rt } (\sigma \text{ nhip}))$ 
   $\wedge \text{nhip} \neq \text{dip}$ 
   $\rightarrow (\text{rt } (\sigma \text{ i})) \sqsubseteq_{\text{dip}} (\text{rt } (\sigma \text{ nhip}))$ )"

(is " $\_ \models (?S \text{ i}, \_ \rightarrow) \_$ ")
proof -
  have weaken:
    " $\bigwedge P \text{ I Q R P}. p \models (\text{otherwith } \text{quality\_increases } \text{I } (\text{orecvmsg } \text{Q}), \text{other } \text{quality\_increases } \text{I } \rightarrow) \text{P}$ 
     $\implies p \models (\text{otherwith } (\text{op=}) \text{I } (\text{orecvmsg } (\lambda\sigma m. \text{Q } \sigma m \wedge \text{R } \sigma m)), \text{other } \text{quality\_increases } \text{I } \rightarrow) \text{P}$ "
    by auto
  {
    fix i a and  $\sigma \sigma' :: \text{"ip} \Rightarrow \text{state"}$ 
    assume a1: " $\forall \text{dip}. \text{dip} \in \text{vD}(\text{rt } (\sigma \text{ i}))$ 
       $\wedge \text{dip} \in \text{vD}(\text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}))))$ 
       $\wedge (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})) \neq \text{dip}$ 
       $\rightarrow \text{rt } (\sigma \text{ i}) \sqsubseteq_{\text{dip}} \text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})))$ "
      and ow: " $?S \text{ i } \sigma \sigma' \text{ a}$ "
    have " $\forall \text{dip}. \text{dip} \in \text{vD}(\text{rt } (\sigma \text{ i}))$ 
       $\wedge \text{dip} \in \text{vD}(\text{rt } (\sigma' (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}))))$ 
       $\wedge (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})) \neq \text{dip}$ 
       $\rightarrow \text{rt } (\sigma \text{ i}) \sqsubseteq_{\text{dip}} \text{rt } (\sigma' (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})))$ "
    proof clarify
      fix dip
      assume a2: " $\text{dip} \in \text{vD}(\text{rt } (\sigma \text{ i}))$ "
      and a3: " $\text{dip} \in \text{vD}(\text{rt } (\sigma' (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}))))$ "
      and a4: " $(\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})) \neq \text{dip}$ "
      from ow have " $\forall j. j \neq i \rightarrow \sigma j = \sigma' j$ " by auto
      show " $\text{rt } (\sigma \text{ i}) \sqsubseteq_{\text{dip}} \text{rt } (\sigma' (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})))$ "
      proof (cases " $(\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})) = i$ ")
        assume " $(\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})) = i$ "
        with  $\langle \text{dip} \in \text{vD}(\text{rt } (\sigma \text{ i})) \rangle$  have " $\text{dip} \in \text{vD}(\text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}))))$ " by simp
        with a1 a2 a4 have " $\text{rt } (\sigma \text{ i}) \sqsubseteq_{\text{dip}} \text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})))$ " by simp
        with  $\langle (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})) = i \rangle$  have " $\text{rt } (\sigma \text{ i}) \sqsubseteq_{\text{dip}} \text{rt } (\sigma \text{ i})$ " by simp
        hence False by simp
        thus ?thesis ..
      next
        assume " $(\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})) \neq i$ "
        with  $\langle \forall j. j \neq i \rightarrow \sigma j = \sigma' j \rangle$ 
        have *: " $\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})) = \sigma' (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}))$ " by simp
        with  $\langle \text{dip} \in \text{vD}(\text{rt } (\sigma' (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})))) \rangle$ 
        have " $\text{dip} \in \text{vD}(\text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}))))$ " by simp
        with a1 a2 a4 have " $\text{rt } (\sigma \text{ i}) \sqsubseteq_{\text{dip}} \text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})))$ " by simp
        with * show ?thesis by simp
      qed
    qed
  }
  note basic = this

  {
    fix  $\sigma \sigma' \text{ a dip sip i}$ 
    assume a1: " $\forall \text{dip}. \text{dip} \in \text{vD}(\text{rt } (\sigma \text{ i}))$ 
       $\wedge \text{dip} \in \text{vD}(\text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}))))$ 
       $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip}) \neq \text{dip}$ 
       $\rightarrow \text{rt } (\sigma \text{ i}) \sqsubseteq_{\text{dip}} \text{rt } (\sigma (\text{the } (\text{nhop } (\text{rt } (\sigma \text{ i})) \text{ dip})))$ "
      and ow: " $?S \text{ i } \sigma \sigma' \text{ a}$ "
    have " $\forall \text{dip}. \text{dip} \in \text{vD}(\text{update } (\text{rt } (\sigma \text{ i})) \text{ sip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \}))$ 
       $\wedge \text{dip} \in \text{vD}(\text{rt } (\sigma' (\text{the } (\text{nhop } (\text{update } (\text{rt } (\sigma \text{ i})) \text{ sip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \})) \text{ dip}))))$ 
       $\wedge \text{the } (\text{nhop } (\text{update } (\text{rt } (\sigma \text{ i})) \text{ sip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \})) \text{ dip}) \neq \text{dip}$ 
       $\rightarrow \text{update } (\text{rt } (\sigma \text{ i})) \text{ sip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \})$ 
       $\sqsubseteq_{\text{dip}} \text{rt } (\sigma' (\text{the } (\text{nhop } (\text{update } (\text{rt } (\sigma \text{ i})) \text{ sip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \})) \text{ dip})))$ "
    proof clarify
      fix dip
      assume a2: " $\text{dip} \in \text{vD}(\text{update } (\text{rt } (\sigma \text{ i})) \text{ sip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \}))$ "
      and a3: " $\text{dip} \in \text{vD}(\text{rt } (\sigma' (\text{the } (\text{nhop } (\text{update } (\text{rt } (\sigma \text{ i})) \text{ sip } (0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}, \{ \})) \text{ dip}))))$ "

```

```

      (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, { }) dip))))"
    and a4: "the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, { }) dip) ≠ dip)"
  show "update (rt (σ i)) sip (0, unk, val, Suc 0, sip, { })
    ⊆dip rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, { }) dip))))"
  proof (cases "dip = sip")
    assume "dip = sip"
    with (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, { }) dip) ≠ dip)
    have False by simp
    thus ?thesis ..
  next
    assume [simp]: "dip ≠ sip"
    from a2 have "dip ∈ vD(rt (σ i)) ∨ dip = sip"
      by (rule vD_update_val)
    with (dip ≠ sip) have "dip ∈ vD(rt (σ i))" by simp
    moreover from a3 have "dip ∈ vD(rt (σ' (the (nhop (rt (σ i)) dip))))" by simp
    moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
    ultimately have "rt (σ i) ⊆dip rt (σ' (the (nhop (rt (σ i)) dip)))"
      using a1 ow by - (drule(1) basic, simp)
    with (dip ≠ sip) show ?thesis
      by - (erule rt_strictly_fresher_update_other, simp)
  qed
} note update_0_unk = this

{ fix σ a σ' nhop
  assume pre: "∀dip. dip ∈ vD(rt (σ i)) ∧ dip ∈ vD(rt (σ (nhop dip))) ∧ nhop dip ≠ dip
    → rt (σ i) ⊆dip rt (σ (nhop dip))"
    and ow: "?S i σ σ' a"
  have "∀dip. dip ∈ vD (invalidate (rt (σ i)) (dests (σ i)))
    ∧ dip ∈ vD (rt (σ' (nhop dip))) ∧ nhop dip ≠ dip
    → rt (σ i) ⊆dip rt (σ' (nhop dip))"
  proof clarify
    fix dip
    assume "dip ∈ vD(invalidate (rt (σ i)) (dests (σ i)))"
      and "dip ∈ vD(rt (σ' (nhop dip)))"
      and "nhop dip ≠ dip"
    from this(1) have "dip ∈ vD (rt (σ i))"
      by (clarsimp dest!: vD_invalidate_vD_not_dests)
    moreover from ow have "∀j. j ≠ i → σ j = σ' j" by auto
    ultimately have "rt (σ i) ⊆dip rt (σ (nhop dip))"
      using pre (dip ∈ vD (rt (σ' (nhop dip))) (nhop dip ≠ dip)
      by metis
    with (∀j. j ≠ i → σ j = σ' j) show "rt (σ i) ⊆dip rt (σ' (nhop dip))"
      by (metis rt_strictly_fresher_irefl)
  qed
} note invalidate = this

{ fix σ a σ' dip oip osn sip hops i
  assume pre: "∀dip. dip ∈ vD (rt (σ i))
    ∧ dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))"
    and ow: "?S i σ σ' a"
    and "Suc 0 ≤ osn"
    and a6: "sip ≠ oip → oip ∈ kD (rt (σ sip))
      ∧ osn ≤ nsqn (rt (σ sip)) oip
      ∧ (nsqn (rt (σ sip)) oip = osn
        → the (dhops (rt (σ sip)) oip) ≤ hops
          ∨ the (flag (rt (σ sip)) oip) = inv)"
  and after: "σ' i = σ i (|rt := update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })|)"
  have "∀dip. dip ∈ vD (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { })
    ∧ dip ∈ vD (rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip, { }) dip))))
    ∧ the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, { }) dip) ≠ dip"

```

```

    → update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})
      □dip
      rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip))))"
proof clarify
fix dip
assume a2: "dip∈vD(update (rt (σ i)) oip (osn, kno, val, Suc (hops), sip, {}))"
and a3: "dip∈vD(rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip, {})) dip))))"
and a4: "the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip"
from ow have a5: "∀j. j ≠ i → σ j = σ' j" by auto
show "update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"
  □dip
  rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip))))"
(is "?rt1 □dip ?rt2 dip")
proof (cases "?rt1 = rt (σ i)")
  assume nochange [simp]:
    "update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {}) = rt (σ i)"

from after have "σ' i = σ i" by simp
with a5 have "∀j. σ j = σ' j" by metis

from a2 have "dip∈vD (rt (σ i))" by simp
moreover from a3 have "dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))"
  using nochange and ⟨∀j. σ j = σ' j⟩ by clarsimp
moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) □dip rt (σ (the (nhop (rt (σ i)) dip)))"
  using pre by simp

hence "rt (σ i) □dip rt (σ' (the (nhop (rt (σ i)) dip)))"
  using ⟨∀j. σ j = σ' j⟩ by simp
thus "?thesis" by simp
next
assume change: "?rt1 ≠ rt (σ i)"
from after a2 have "dip∈kD(rt (σ' i))" by auto
show ?thesis
proof (cases "dip = oip")
  assume "dip ≠ oip"

  with a2 have "dip∈vD (rt (σ i))" by auto
  moreover with a3 a5 after and ⟨dip ≠ oip⟩
    have "dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))"
      by simp metis
  moreover from a4 and ⟨dip ≠ oip⟩ have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
  ultimately have "rt (σ i) □dip rt (σ (the (nhop (rt (σ i)) dip)))"
    using pre by simp

  with after and a5 and ⟨dip ≠ oip⟩ show ?thesis
    by simp (metis rt_strictly_fresher_update_other
      rt_strictly_fresher_irefl)
next
assume "dip = oip"

with a4 and change have "sip ≠ oip" by simp
with a6 have "oip∈kD(rt (σ sip))"
  and "osn ≤ nsqn (rt (σ sip)) oip" by auto

from a3 change ⟨dip = oip⟩ have "oip∈vD(rt (σ' sip))" by simp
hence "the (flag (rt (σ' sip)) oip) = val" by simp

from ⟨oip∈kD(rt (σ sip))⟩
have "osn < nsqn (rt (σ' sip)) oip ∨ (osn = nsqn (rt (σ' sip)) oip
  ∧ the (dhops (rt (σ' sip)) oip) ≤ hops)"
proof
  assume "oip∈vD(rt (σ sip))"

```

```

hence "the (flag (rt (σ sip)) oip) = val" by simp
with a6 ⟨sip ≠ oip⟩ have "nsqn (rt (σ sip)) oip = osn →
      the (dhops (rt (σ sip)) oip) ≤ hops"

  by simp
show ?thesis
proof (cases "sip = i")
  assume "sip ≠ i"
  with a5 have "σ sip = σ' sip" by simp
  with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
  and ⟨nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
  show ?thesis by auto
next
— alternative to using sip_not_ip
assume [simp]: "sip = i"
have "?rt1 = rt (σ i)"
proof (rule update_cases_kD, simp_all)
  from ⟨Suc 0 ≤ osn⟩ show "0 < osn" by simp
next
  from ⟨oip ∈ kD(rt (σ sip))⟩ and ⟨sip = i⟩ show "oip ∈ kD(rt (σ i))"
  by simp
next
  assume "sqn (rt (σ i)) oip < osn"
  also from ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
  have "... ≤ nsqn (rt (σ i)) oip" by simp
  also have "... ≤ sqn (rt (σ i)) oip"
  by (rule nsqn_sqn)
  finally have "sqn (rt (σ i)) oip < sqn (rt (σ i)) oip" .
  hence False by simp
  thus "(λa. if a = oip
    then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
    else rt (σ i) a) = rt (σ i)" ..
next
  assume "sqn (rt (σ i)) oip = osn"
  and "Suc hops < the (dhops (rt (σ i)) oip)"
  from this(1) and ⟨oip ∈ vD (rt (σ sip))⟩ have "nsqn (rt (σ i)) oip = osn"
  by simp
  with ⟨nsqn (rt (σ sip)) oip = osn → the (dhops (rt (σ sip)) oip) ≤ hops⟩
  have "the (dhops (rt (σ i)) oip) ≤ hops" by simp
  with ⟨Suc hops < the (dhops (rt (σ i)) oip)⟩ have False by simp
  thus "(λa. if a = oip
    then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
    else rt (σ i) a) = rt (σ i)" ..
next
  assume "the (flag (rt (σ i)) oip) = inv"
  with ⟨the (flag (rt (σ sip)) oip) = val⟩ have False by simp
  thus "(λa. if a = oip
    then Some (osn, kno, val, Suc hops, i, π7 (the (rt (σ i) oip)))
    else rt (σ i) a) = rt (σ i)" ..
next
  from ⟨oip ∈ kD(rt (σ sip))⟩
  show "(λa. if a = oip then Some (the (rt (σ i) oip)) else rt (σ i) a) = rt (σ i)"
  by (auto dest!: kD_Some)
qed
with change have False ..
thus ?thesis ..
qed
next
assume "oip ∈ iD(rt (σ sip))"
with ⟨the (flag (rt (σ' sip)) oip) = val⟩ and a5 have "sip = i"
  by (metis f.distinct(1) iD_flag_is_inv)
from ⟨oip ∈ iD(rt (σ sip))⟩ have "the (flag (rt (σ sip)) oip) = inv" by auto
with ⟨sip = i⟩ ⟨Suc 0 ≤ osn⟩ change after ⟨oip ∈ kD(rt (σ sip))⟩
  have "nsqn (rt (σ sip)) oip < nsqn (rt (σ' sip)) oip"
  unfolding update_def

```

```

      by (clarsimp split: option.split_asm split_if_asm)
        (auto simp: sqn_def)
with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩ have "osn < nsqn (rt (σ' sip)) oip"
  by simp
thus ?thesis ..
qed
thus ?thesis
proof
  assume osnlt: "osn < nsqn (rt (σ' sip)) oip"
  from ⟨dip ∈ kD(rt (σ' i))⟩ and ⟨dip = oip⟩ have "dip ∈ kD (?rt1)" by simp
  moreover from a3 have "dip ∈ kD(?rt2 dip)" by simp
  moreover have "nsqn ?rt1 dip < nsqn (?rt2 dip) dip"
    proof -
      have "nsqn ?rt1 oip = osn"
        by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
      also have "... < nsqn (rt (σ' sip)) oip" using osnlt .
      also have "... = nsqn (?rt2 oip) oip" by (simp add: change)
      finally show ?thesis
        using ⟨dip = oip⟩ by simp
    qed
  ultimately show ?thesis
    by (rule rt_strictly_fresher_ltI)
next
  assume osneq: "osn = nsqn (rt (σ' sip)) oip ∧ the (dhops (rt (σ' sip)) oip) ≤ hops"

  have "oip ∈ kD(?rt1)" by simp
  moreover from a3 ⟨dip = oip⟩ have "oip ∈ kD(?rt2 oip)" by simp

  moreover have "nsqn ?rt1 oip = nsqn (?rt2 oip) oip"
  proof -
    from osneq have "osn = nsqn (rt (σ' sip)) oip" ..
    also have "osn = nsqn ?rt1 oip"
      by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
    also have "nsqn (rt (σ' sip)) oip = nsqn (?rt2 oip) oip"
      by (simp add: change)
    finally show ?thesis .
  qed

  moreover have "π5(the (?rt2 oip oip)) < π5(the (?rt1 oip))"
  proof -
    from osneq have "the (dhops (rt (σ' sip)) oip) ≤ hops" ..
    moreover from ⟨oip ∈ vD (rt (σ' sip))⟩ have "oip ∈ kD(rt (σ' sip))" by auto
    ultimately have "π5(the (rt (σ' sip) oip)) ≤ hops"
      by (auto simp add: proj5_eq_dhops)
    also from change after have "hops < π5(the (rt (σ' i) oip))"
      by (simp add: proj5_eq_dhops) (metis dhops_update_changed lessI)
    finally have "π5(the (rt (σ' sip) oip)) < π5(the (rt (σ' i) oip))" .
    with change after show ?thesis by simp
  qed

  ultimately have "?rt1 □oip ?rt2 oip"
    by (rule rt_strictly_fresher_eqI)
  with ⟨dip = oip⟩ show ?thesis by simp
  qed
qed
qed
qed
} note rreq_rrep_update = this

have "opaadv i ⊨ (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
                                                ∧ msg_zhops m))),
      other quality_increases {i} →)
  onl ΓAODV
  (λ(σ, _). ∀dip. dip ∈ vD (rt (σ i)) ∩ vD (rt (σ (the (nhop (rt (σ i)) dip))))"

```

```

      ∧ the (nhop (rt (σ i)) dip) ≠ dip
      → rt (σ i) ⊑dip rt (σ (the (nhop (rt (σ i)) dip))))"
proof (inv_cterms inv add: onl_oinvariant_sterms [OF aadv_wf rreq_sip [THEN weaken]]
      onl_oinvariant_sterms [OF aadv_wf rrep_sip [THEN weaken]]
      onl_oinvariant_sterms [OF aadv_wf rerr_sip [THEN weaken]]
      onl_oinvariant_sterms [OF aadv_wf oosn_rreq [THEN weaken]]
      onl_oinvariant_sterms [OF aadv_wf odsn_rrep [THEN weaken]]
      onl_oinvariant_sterms [OF aadv_wf oadpreRT_welldefined])
      solve: basic update_0_unk invalidate rreq_rrep_update
      simp add: seqsimp)
fix σ σ' p l
assume or: "(σ, p) ∈ oreachable (opaadv i) (?S i) (other quality_increases {i})"
  and "other quality_increases {i} σ σ'"
  and ll: "l ∈ labels ΓAADV p"
  and pre: "∀dip. dip ∈ vD (rt (σ i))
    ∧ dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) ⊑dip rt (σ (the (nhop (rt (σ i)) dip)))"
from this(1-2)
  have or': "(σ', p) ∈ oreachable (opaadv i) (?S i) (other quality_increases {i})"
    by - (rule oreachable_other')

from or and ll have next_hop: "∀dip. let nhip = the (nhop (rt (σ i)) dip)
  in dip ∈ kD (rt (σ i)) ∧ nhip ≠ dip
  → dip ∈ kD (rt (σ nhip))
  ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip"
  by (auto dest!: onl_oinvariant_weakenD [OF seq_compare_next_hop'])

from or and ll have unk_hops_one: "∀dip ∈ kD (rt (σ i)). sqn (rt (σ i)) dip = 0
  → sqnf (rt (σ i)) dip = unk
  ∧ the (dhops (rt (σ i)) dip) = 1
  ∧ the (nhop (rt (σ i)) dip) = dip"
  by (auto dest!: onl_oinvariant_weakenD [OF ozero_seq_unk_hops_one
    [OF oaadv_trans aadv_trans]]
    otherwith_actionD
    simp: seqsimp)

from ⟨other quality_increases {i} σ σ'⟩ have "σ' i = σ i" by auto
hence "quality_increases (σ i) (σ' i)" by auto
with ⟨other quality_increases {i} σ σ'⟩ have "∀j. quality_increases (σ j) (σ' j)"
  by - (erule otherE, metis singleton_iff)

show "∀dip. dip ∈ vD (rt (σ' i))
  ∧ dip ∈ vD (rt (σ' (the (nhop (rt (σ' i)) dip))))
  ∧ the (nhop (rt (σ' i)) dip) ≠ dip
  → rt (σ' i) ⊑dip rt (σ' (the (nhop (rt (σ' i)) dip)))"
proof clarify
  fix dip
  assume "dip ∈ vD (rt (σ' i))"
  and "dip ∈ vD (rt (σ' (the (nhop (rt (σ' i)) dip))))"
  and "the (nhop (rt (σ' i)) dip) ≠ dip"
  from this(1) and ⟨σ' i = σ i⟩ have "dip ∈ vD (rt (σ i))"
    and "dip ∈ kD (rt (σ i))"
  by auto

  from ⟨the (nhop (rt (σ' i)) dip) ≠ dip⟩ and ⟨σ' i = σ i⟩
  have "the (nhop (rt (σ i)) dip) ≠ dip" (is "?nhip ≠ _") by simp
  with ⟨dip ∈ kD (rt (σ i))⟩ and next_hop
  have "dip ∈ kD (rt (σ (?nhip)))"
  and nsqns: "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ ?nhip)) dip"
  by (auto simp: Let_def)

  have "0 < sqn (rt (σ i)) dip"
  proof (rule neq0_conv [THEN iffD1, OF notI])

```



```

    assume "sqn (rt (σ i)) dip = 0"
    with ⟨dip∈kD(rt (σ i))⟩ and unk_hops_one
      have "?nhip = dip" by simp
    with ⟨?nhip ≠ dip⟩ show False ..
  qed
also have "... = nsqn (rt (σ i)) dip"
  by (rule vD_nsqn_sqn [OF ⟨dip∈vD(rt (σ i))⟩, THEN sym])
also have "... ≤ nsqn (rt (σ ?nhip)) dip"
  by (rule nsqns)
also have "... ≤ sqn (rt (σ ?nhip)) dip"
  by (rule nsqn_sqn)
finally have "0 < sqn (rt (σ ?nhip)) dip" .

have "rt (σ i) ⊑dip rt (σ' ?nhip)"
proof (cases "dip∈vD(rt (σ ?nhip))")
  assume "dip∈vD(rt (σ ?nhip))"
  with pre ⟨dip∈vD(rt (σ i))⟩ and ⟨?nhip ≠ dip⟩
    have "rt (σ i) ⊑dip rt (σ ?nhip)" by auto
  moreover from ⟨∀j. quality_increases (σ j) (σ' j)⟩
    have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
  ultimately show ?thesis
    using ⟨dip∈kD(rt (σ ?nhip))⟩
      by (rule strictly_fresher_quality_increases_right)
next
  assume "dip∉vD(rt (σ ?nhip))"
  with ⟨dip∈kD(rt (σ ?nhip))⟩ have "dip∈iD(rt (σ ?nhip))" ..
  hence "the (flag (rt (σ ?nhip)) dip) = inv"
    by auto
  have "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ ?nhip)) dip"
    by (rule nsqns)
  also from ⟨dip∈iD(rt (σ ?nhip))⟩
    have "... = sqn (rt (σ ?nhip)) dip - 1" ..
  also have "... < sqn (rt (σ' ?nhip)) dip"
    proof -
      from ⟨∀j. quality_increases (σ j) (σ' j)⟩
        have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
      hence "∀ip. sqn (rt (σ ?nhip)) ip ≤ sqn (rt (σ' ?nhip)) ip" by auto
      hence "sqn (rt (σ ?nhip)) dip ≤ sqn (rt (σ' ?nhip)) dip" ..
      with ⟨0 < sqn (rt (σ ?nhip)) dip⟩ show ?thesis by auto
    qed
  also have "... = nsqn (rt (σ' ?nhip)) dip"
    proof (rule vD_nsqn_sqn [THEN sym])
      from ⟨dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and ⟨σ' i = σ i⟩
        show "dip∈vD(rt (σ' ?nhip))" by simp
    qed
  finally have "nsqn (rt (σ i)) dip < nsqn (rt (σ' ?nhip)) dip" .

  moreover from ⟨dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and ⟨σ' i = σ i⟩
    have "dip∈kD(rt (σ' ?nhip))" by auto
  ultimately show "rt (σ i) ⊑dip rt (σ' ?nhip)"
    using ⟨dip∈kD(rt (σ i))⟩ by - (rule rt_strictly_fresher_ltI)
  qed
with ⟨σ' i = σ i⟩ show "rt (σ' i) ⊑dip rt (σ' (the (nhop (rt (σ' i)) dip)))"
  by simp
qed
qed
thus ?thesis unfolding Let_def .
qed

```

lemma seq\_compare\_next\_hop:

fixes w

shows "opaodv i ⊨ (otherwith (op=) {i} (orecvmsg msg\_fresh),  
 other quality\_increases {i} →)  
 global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)

```

      in dip ∈ kD(rt (σ i)) ∧ nhip ≠ dip →
      dip ∈ kD(rt (σ nhip))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip)"
by (rule oinvariant_weakenE [OF seq_compare_next_hop']) (auto dest!: onlD)

```

**lemma seq\_nhop\_quality\_increases:**

```

shows "opaadv i ⊨ (otherwith (op=) {i}
  (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
  other quality_increases {i} →)
  global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊑dip (rt (σ nhip)))"
by (rule oinvariant_weakenE [OF seq_nhop_quality_increases']) (auto dest!: onlD)

```

end

## 4.10 Routing graphs and loop freedom

```

theory D_Loop_Freedom
imports D_Aodv_Predicates D_Fresher
begin

```

Define the central theorem that relates an invariant over network states to the absence of loops in the associate routing graph.

**definition**

```

rt_graph :: "(ip ⇒ state) ⇒ ip ⇒ ip rel"
where
  "rt_graph σ = (λdip.
    {(ip, ip') | ip ip' dsn dsk hops pre.
      ip ≠ dip ∧ rt (σ ip) dip = Some (dsn, dsk, val, hops, ip', pre)})"

```

Given the state of a network  $\sigma$ , a routing graph for a given destination  $dip$  abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

**lemma rt\_graphE [elim]:**

```

fixes n dip ip ip'
assumes "(ip, ip') ∈ rt_graph σ dip"
shows "ip ≠ dip ∧ (∃r. rt (σ ip) = r
  ∧ (∃dsn dsk hops pre. r dip = Some (dsn, dsk, val, hops, ip', pre)))"
using assms unfolding rt_graph_def by auto

```

**lemma rt\_graph\_vD [dest]:**

```

"∧ip ip' σ dip. (ip, ip') ∈ rt_graph σ dip ⇒ dip ∈ vD(rt (σ ip))"
unfolding rt_graph_def vD_def by auto

```

**lemma rt\_graph\_vD\_trans [dest]:**

```

"∧ip ip' σ dip. (ip, ip') ∈ (rt_graph σ dip)+ ⇒ dip ∈ vD(rt (σ ip))"
by (erule converse_tranclE) auto

```

**lemma rt\_graph\_not\_dip [dest]:**

```

"∧ip ip' σ dip. (ip, ip') ∈ rt_graph σ dip ⇒ ip ≠ dip"
unfolding rt_graph_def by auto

```

**lemma rt\_graph\_not\_dip\_trans [dest]:**

```

"∧ip ip' σ dip. (ip, ip') ∈ (rt_graph σ dip)+ ⇒ ip ≠ dip"
by (erule converse_tranclE) auto

```

NB: the property below cannot be lifted to the transitive closure

**lemma rt\_graph\_nhip\_is\_nhop [dest]:**

```

"∧ip ip' σ dip. (ip, ip') ∈ rt_graph σ dip ⇒ ip' = the (nhop (rt (σ ip)) dip)"
unfolding rt_graph_def by auto

```

**theorem inv\_to\_loop\_freedom:**

```

assumes "∀i dip. let nhip = the (nhop (rt (σ i)) dip)

```

```

      in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
      → (rt (σ i)) ⊆dip (rt (σ nhip))"
shows "∀dip. irrefl ((rt_graph σ dip)+)"
using assms proof (intro allI)
fix σ :: "ip ⇒ state" and dip
assume inv: "∀ip dip.
  let nhip = the (nhop (rt (σ ip)) dip)
  in dip ∈ vD(rt (σ ip)) ∩ vD(rt (σ nhip)) ∧
  nhip ≠ dip → rt (σ ip) ⊆dip rt (σ nhip)"
{ fix ip ip'
  assume "(ip, ip') ∈ (rt_graph σ dip)+"
  and "dip ∈ vD(rt (σ ip'))"
  and "ip' ≠ dip"
  hence "rt (σ ip) ⊆dip rt (σ ip')"
  proof induction
    fix nhip
    assume "(ip, nhip) ∈ rt_graph σ dip"
    and "dip ∈ vD(rt (σ nhip))"
    and "nhip ≠ dip"
    from ⟨(ip, nhip) ∈ rt_graph σ dip⟩ have "dip ∈ vD(rt (σ ip))"
    and "nhip = the (nhop (rt (σ ip)) dip)"

    by auto
    from ⟨dip ∈ vD(rt (σ ip))⟩ and ⟨dip ∈ vD(rt (σ nhip))⟩
    have "dip ∈ vD(rt (σ ip)) ∩ vD(rt (σ nhip))" ..
    with ⟨nhip = the (nhop (rt (σ ip)) dip)⟩
    and ⟨nhip ≠ dip⟩
    and inv
    show "rt (σ ip) ⊆dip rt (σ nhip)"
    by (clarsimp simp: Let_def)
  next
    fix nhip nhip'
    assume "(ip, nhip) ∈ (rt_graph σ dip)+"
    and "(nhip, nhip') ∈ rt_graph σ dip"
    and IH: "[[ dip ∈ vD(rt (σ nhip)); nhip ≠ dip ]] ⇒ rt (σ ip) ⊆dip rt (σ nhip)"
    and "dip ∈ vD(rt (σ nhip'))"
    and "nhip' ≠ dip"
    from ⟨(nhip, nhip') ∈ rt_graph σ dip⟩ have 1: "dip ∈ vD(rt (σ nhip))"
    and 2: "nhip ≠ dip"
    and "nhip = the (nhop (rt (σ nhip)) dip)"

    by auto
    from 1 2 have "rt (σ ip) ⊆dip rt (σ nhip)" by (rule IH)
    also have "rt (σ nhip) ⊆dip rt (σ nhip')"
    proof -
      from ⟨dip ∈ vD(rt (σ nhip))⟩ and ⟨dip ∈ vD(rt (σ nhip'))⟩
      have "dip ∈ vD(rt (σ nhip)) ∩ vD(rt (σ nhip'))" ..
      with ⟨nhip' ≠ dip⟩
      and ⟨nhip = the (nhop (rt (σ nhip)) dip)⟩
      and inv
      show "rt (σ nhip) ⊆dip rt (σ nhip')"
      by (clarsimp simp: Let_def)
    qed
    finally show "rt (σ ip) ⊆dip rt (σ nhip')" .
  qed } note fresher = this

show "irrefl ((rt_graph σ dip)+)"
unfolding irrefl_def proof (intro allI notI)
fix ip
assume "(ip, ip) ∈ (rt_graph σ dip)+"
moreover then have "dip ∈ vD(rt (σ ip))"
and "ip ≠ dip"

by auto
ultimately have "rt (σ ip) ⊆dip rt (σ ip)" by (rule fresher)
thus False by simp
qed

```

qed

end

## 4.11 Lift and transfer invariants to show loop freedom

theory *D\_Aodv\_Loop\_Freedom*

imports "../.../AWN/OClosed\_Transfer" "../.../AWN/Qmsg\_Lifting" *D\_Global\_Invariants* *D\_Loop\_Freedom*  
begin

### 4.11.1 Lift to parallel processes with queues

lemma *par\_step\_no\_change\_on\_send\_or\_receive*:

fixes  $\sigma$   $s$   $a$   $\sigma'$   $s'$   
assumes " $((\sigma, s), a, (\sigma', s')) \in \text{oparp\_sos } i \text{ (oseqp\_sos } \Gamma_{AODV} i) \text{ (seqp\_sos } \Gamma_{QMSG})$ "  
and " $a \neq \tau$ "  
shows " $\sigma' i = \sigma i$ "  
using *assms* by (rule *qmsg\_no\_change\_on\_send\_or\_receive*)

lemma *par\_nhop\_quality\_increases*:

shows " $\text{opaodv } i \ll_i \text{ qmsg} \models (\text{otherwith } (op=) \{i\} (\text{orecvmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)), \text{other quality\_increases } \{i\} \rightarrow)$   
global  $(\lambda \sigma. \forall \text{dip. let } nhop = \text{the } (nhop \text{ (rt } (\sigma i)) \text{ dip})$   
in  $\text{dip} \in \text{vD } (\text{rt } (\sigma i)) \cap \text{vD } (\text{rt } (\sigma nhop)) \wedge nhop \neq \text{dip}$   
 $\rightarrow (\text{rt } (\sigma i)) \sqsubseteq_{\text{dip}} (\text{rt } (\sigma nhop))$ )"

proof (rule *lift\_into\_qmsg* [OF *seq\_nhop\_quality\_increases*])

show " $\text{opaodv } i \models_A (\text{otherwith } (op=) \{i\} (\text{orecvmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)), \text{other quality\_increases } \{i\} \rightarrow)$   
globala  $(\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

proof (rule *ostep\_invariant\_weakenE* [OF *quality\_increases*], *simp\_all*)

fix  $t$  :: " $((\text{nat} \Rightarrow \text{state}) \times (\text{state}, \text{msg}, \text{pseqp}, \text{pseqp label}) \text{ seqp}), \text{msg seq\_action}) \text{ transition}$ "  
assume " $\text{onll } \Gamma_{AODV} (\lambda((\sigma, \_), \_, (\sigma', \_)). \forall j. \text{quality\_increases } (\sigma j) (\sigma' j)) t$ "  
thus " $\text{quality\_increases } (\text{fst } (\text{fst } t) i) (\text{fst } (\text{snd } (\text{snd } t)) i)$ "  
by (*cases*  $t$ ) (*clarsimp* *dest!*: *onllD*, *metis* *aodv\_ex\_label*)

next

fix  $\sigma$   $\sigma'$   $a$   
assume " $\text{otherwith } (op=) \{i\} (\text{orecvmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)) \sigma \sigma' a$ "  
thus " $\text{otherwith quality\_increases } \{i\} (\text{orecvmsg } (\lambda \_. \text{rreq\_rrep\_sn})) \sigma \sigma' a$ "  
by - (*erule* *weaken\_otherwith*, *auto*)

qed

qed *auto*

lemma *par\_rreq\_rrep\_sn\_quality\_increases*:

" $\text{opaodv } i \ll_i \text{ qmsg} \models_A (\lambda \sigma \_. \text{orecvmsg } (\lambda \_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda \_ \_. \text{True}) \{i\} \rightarrow)$   
globala  $(\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

proof -

have " $\text{opaodv } i \models_A (\lambda \sigma \_. \text{orecvmsg } (\lambda \_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda \_ \_. \text{True}) \{i\} \rightarrow)$   
globala  $(\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

by (rule *ostep\_invariant\_weakenE* [OF *olocal\_quality\_increases*])  
(*auto* *dest!*: *onllD* *seqllD* *elim!*: *aodv\_ex\_labelE*)

hence " $\text{opaodv } i \ll_i \text{ qmsg} \models_A (\lambda \sigma \_. \text{orecvmsg } (\lambda \_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda \_ \_. \text{True}) \{i\} \rightarrow)$   
globala  $(\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

by (rule *lift\_step\_into\_qmsg\_statelessassm*) *simp\_all*

thus ?*thesis* by rule *auto*

qed

lemma *par\_rreq\_rrep\_nsqn\_fresh\_any\_step*:

shows " $\text{opaodv } i \ll_i \text{ qmsg} \models_A (\lambda \sigma \_. \text{orecvmsg } (\lambda \_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda \_ \_. \text{True}) \{i\} \rightarrow)$   
globala  $(\lambda(\sigma, a, \sigma'). \text{anycast } (\text{msg\_fresh } \sigma) a)$ "

proof -

```

have "opaadv i  $\models_A (\lambda\sigma \_ . (\text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn})) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\sigma, a, \sigma') . \text{anycast } (\text{msg\_fresh } \sigma) a)$ "
proof (rule ostep_invariant_weakenE [OF rreq_rrep_nsqn_fresh_any_step_invariant])
  fix t
  assume "onll  $\Gamma_{AODV} (\lambda((\sigma, \_), a, \_). \text{anycast } (\text{msg\_fresh } \sigma) a) t$ "
  thus "globala  $(\lambda(\sigma, a, \sigma') . \text{anycast } (\text{msg\_fresh } \sigma) a) t$ "
    by (cases t) (clarsimp dest!: onllD, metis aadv_ex_label)
qed auto
hence "opaadv i  $\langle\langle_i \text{qmsg } \models_A (\lambda\sigma \_ . (\text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn})) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\sigma, a, \sigma') . \text{anycast } (\text{msg\_fresh } \sigma) a)$ "
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

```

lemma par\_anycast\_msg\_zhops:

```

shows "opaadv i  $\langle\langle_i \text{qmsg } \models_A (\lambda\sigma \_ . \text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a)$ "

```

proof -

```

from anycast_msg_zhops initiali_aadv oaadv_trans aadv_trans

```

```

have "opaadv i  $\models_A (\text{act } TT, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      seqll i (onll  $\Gamma_{AODV} (\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a)$ )"
  by (rule open_seq_step_invariant)

```

```

hence "opaadv i  $\models_A (\lambda\sigma \_ . \text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a)$ "

```

proof (rule ostep\_invariant\_weakenE)

```

fix t :: "((nat  $\Rightarrow$  state)  $\times$  (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"

```

```

assume "seqll i (onll  $\Gamma_{AODV} (\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a)$ ) t"

```

```

thus "globala  $(\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a) t$ "

```

```

  by (cases t) (clarsimp dest!: seqllD onllD, metis aadv_ex_label)

```

qed simp\_all

```

hence "opaadv i  $\langle\langle_i \text{qmsg } \models_A (\lambda\sigma \_ . \text{orecvmsg } (\lambda\_ . \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\_, a, \_). \text{anycast } \text{msg\_zhops } a)$ "

```

```

  by (rule lift_step_into_qmsg_statelessassm) simp_all

```

```

thus ?thesis by rule auto

```

qed

#### 4.11.2 Lift to nodes

lemma node\_step\_no\_change\_on\_send\_or\_receive:

```

assumes "(( $\sigma, \text{NodeS } i \text{ P } R$ ), a, ( $\sigma', \text{NodeS } i' \text{ P}' R'$ ))  $\in$  onode_sos
      (oparp_sos i (oseqp_sos  $\Gamma_{AODV}$  i) (seqp_sos  $\Gamma_{QMSG}$ ))"

```

```

  and "a  $\neq \tau$ "

```

```

  shows " $\sigma' i = \sigma i$ "

```

using assms

```

by (cases a) (auto elim!: par_step_no_change_on_send_or_receive)

```

lemma node\_nhop\_quality\_increases:

```

shows "< i : opaadv i  $\langle\langle_i \text{qmsg} : R \rangle_o \models$ 
      (otherwith (op=) {i}
        (oarrivmsg  $(\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)$ ),
        other quality_increases {i}
       $\rightarrow$  global  $(\lambda\sigma. \forall \text{dip. let } \text{nhop} = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip})$ 
        in  $\text{dip} \in \text{vD } (\text{rt } (\sigma i)) \cap \text{vD } (\text{rt } (\sigma \text{nhop})) \wedge \text{nhop} \neq \text{dip}$ 
         $\rightarrow (\text{rt } (\sigma i)) \sqsubseteq_{\text{dip}} (\text{rt } (\sigma \text{nhop}))$ )"
  by (rule node_lift [OF par_nhop_quality_increases]) auto

```

lemma node\_quality\_increases:

```

"< i : opaadv i  $\langle\langle_i \text{qmsg} : R \rangle_o \models_A (\lambda\sigma \_ . \text{oarrivmsg } (\lambda\_ . \text{rreq\_rrep\_sn}) \sigma,$ 
      other  $(\lambda\_ \_ . \text{True}) \{i\} \rightarrow)$ 
      globala  $(\lambda(\sigma, \_, \sigma') . \text{quality\_increases } (\sigma i) (\sigma' i))"$ 
  by (rule node_lift_step_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp

```

lemma node\_rreq\_rrep\_nsqn\_fresh\_any\_step:

```

shows "< i : opaadv i  $\langle\langle_i \text{qmsg} : R \rangle_o \models_A$ 

```

```

      (λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
      globala (λ(σ, a, σ'). castmsg (msg_fresh σ) a)"
  by (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_nsqn_fresh_any_step])

```

lemma node\_anycast\_msg\_zhops:

```

  shows "⟨ i : opaadv i ⟨⟨i qmsg : R ⟩o ⟩A
        (λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →)
        globala (λ(_, a, _). castmsg msg_zhops a)"
  by (rule node_lift_anycast_statelessassm [OF par_anycast_msg_zhops])

```

lemma node\_silent\_change\_only:

```

  shows "⟨ i : opaadv i ⟨⟨i qmsg : Ri ⟩o ⟩A (λσ _. oarrivemsg (λ_ _. True) σ,
        other (λ_ _. True) {i} →)
        globala (λ(σ, a, σ'). a ≠ τ → σ' i = σ i)"

```

proof (rule ostep\_invariantI, simp (no\_asm), rule impI)

fix σ ζ a σ' ζ'

```

  assume or: "(σ, ζ) ∈ oreachable (⟨i : opaadv i ⟨⟨i qmsg : Ri ⟩o ⟩
        (λσ _. oarrivemsg (λ_ _. True) σ)
        (other (λ_ _. True) {i}))"

```

```

  and tr: "((σ, ζ), a, (σ', ζ')) ∈ trans (⟨i : opaadv i ⟨⟨i qmsg : Ri ⟩o ⟩)"
  and "a ≠ τn"

```

from or obtain p R where "ζ = NodeS i p R"

by - (drule node\_net\_state, metis)

with tr have "(σ, NodeS i p R), a, (σ', ζ')

```

  ∈ onode_sos (oparp_sos i (trans (opaadv i)) (trans qmsg))"

```

by simp

thus "σ' i = σ i" using ⟨a ≠ τ<sub>n</sub>⟩

by (cases rule: onode\_sos.cases)

(auto elim: qmsg\_no\_change\_on\_send\_or\_receive)

qed

### 4.11.3 Lift to partial networks

lemma arrive\_rreq\_rrep\_nsqn\_fresh\_inc\_sn [simp]:

assumes "oarrivemsg (λσ m. msg\_fresh σ m ∧ P σ m) σ m"

shows "oarrivemsg (λ\_. rreq\_rrep\_sn) σ m"

using assms by (cases m) auto

lemma opnet\_nhop\_quality\_increases:

```

  shows "opnet (λi. opaadv i ⟨⟨i qmsg ⟩ p ⟩
        (otherwith (op=) (net_tree_ips p)
        (oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
        other quality_increases (net_tree_ips p) →)
        global (λσ. ∀i∈net_tree_ips p. ∀dip.
          let nhip = the (nhop (rt (σ i)) dip)
          in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
          → (rt (σ i)) ⊆dip (rt (σ nhip)))"

```

proof (rule pnet\_lift [OF node\_nhop\_quality\_increases])

fix i R

```

  have "⟨i : opaadv i ⟨⟨i qmsg : R ⟩o ⟩A (λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ,
        other (λ_ _. True) {i} →) globala (λ(σ, a, σ').
        castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"

```

proof (rule ostep\_invariantI, simp (no\_asm))

fix σ s a σ' s'

```

  assume or: "(σ, s) ∈ oreachable (⟨i : opaadv i ⟨⟨i qmsg : R ⟩o ⟩
        (λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ)
        (other (λ_ _. True) {i}))"

```

```

  and tr: "((σ, s), a, (σ', s')) ∈ trans (⟨i : opaadv i ⟨⟨i qmsg : R ⟩o ⟩)"

```

```

  and am: "oarrivemsg (λ_. rreq_rrep_sn) σ a"

```

from or tr am have "castmsg (msg\_fresh σ) a"

by (auto dest!: ostep\_invariantD [OF node\_rreq\_rrep\_nsqn\_fresh\_any\_step])

moreover from or tr am have "castmsg (msg\_zhops) a"

by (auto dest!: ostep\_invariantD [OF node\_anycast\_msg\_zhops])

ultimately show "castmsg (λm. msg\_fresh σ m ∧ msg\_zhops m) a"

```

    by (case_tac a) auto
qed
thus "<i : opaadv i <<i qmsg : R>_o ⊢A
      (λσ _. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
      other quality_increases {i} →) globala (λ(σ, a, _).
      castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
  by rule auto
next
fix i R
show "<i : opaadv i <<i qmsg : R>_o ⊢A
      (λσ _. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
      other quality_increases {i} →) globala (λ(σ, a, σ').
      a ≠ τ ∧ (∀i d. a ≠ i:deliver(d)) → σ i = σ' i)"
  by (rule ostep_invariant_weakenE [OF node_silent_change_only]) auto
next
fix i R
show "<i : opaadv i <<i qmsg : R>_o ⊢A
      (λσ _. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
      other quality_increases {i} →) globala (λ(σ, a, σ').
      a = τ ∨ (∃d. a = i:deliver(d)) → quality_increases (σ i) (σ' i))"
  by (rule ostep_invariant_weakenE [OF node_quality_increases]) auto
qed simp_all

```

#### 4.11.4 Lift to closed networks

lemma onet\_nhop\_quality\_increases:

```

shows "oclosed (opnet (λi. opaadv i <<i qmsg> p)
  ⊢ (λ_ _ . True, other quality_increases (net_tree_ips p) →)
  global (λσ. ∀i∈net_tree_ips p. ∀dip.
    let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊆dip (rt (σ nhip)))"

```

(is "\_ ⊢ (\_, ?U →) ?inv")

proof (rule inclosed\_closed)

from opnet\_nhop\_quality\_increases

```

show "opnet (λi. opaadv i <<i qmsg> p
  ⊢ (otherwith (op=) (net_tree_ips p) inclosed, ?U →) ?inv"

```

proof (rule oinvariant\_weakenE)

fix σ σ' :: "ip ⇒ state" and a :: "msg node\_action"

assume "otherwith (op=) (net\_tree\_ips p) inclosed σ σ' a"

thus "otherwith (op=) (net\_tree\_ips p)

(oarrivemsg (λσ m. msg\_fresh σ m ∧ msg\_zhops m)) σ σ' a"

proof (rule otherwithEI)

fix σ :: "ip ⇒ state" and a :: "msg node\_action"

assume "inclosed σ a"

thus "oarrivemsg (λσ m. msg\_fresh σ m ∧ msg\_zhops m) σ a"

proof (cases a)

fix ii ni ms

assume "a = ii-ni:arrive(ms)"

moreover with (inclosed σ a) obtain d di where "ms = newpkt(d, di)"

by (cases ms) auto

ultimately show ?thesis by simp

qed simp\_all

qed

qed

qed

#### 4.11.5 Transfer into the standard model

interpretation aadv\_openproc: openproc paadv opaadv id

where "aadv\_openproc.initmissing = initmissing"

proof -

show "openproc paadv opaadv id"

proof unfold\_locales

```

fix i :: ip
have "{(σ, ζ). (σ i, ζ) ∈ σAODV i ∧ (∀j. j ≠ i → σ j ∈ fst ' σAODV j)} ⊆ σAODV'"
  unfolding σAODV_def σAODV'_def
  proof (rule equalityD1)
    show "∧f p. {(σ, ζ). (σ i, ζ) ∈ {(f i, p)} ∧ (∀j. j ≠ i
      → σ j ∈ fst ' {(f j, p)}')} = {(f, p)}"
      by (rule set_eqI) auto
  qed
thus "{(σ, ζ) | σ ζ s. s ∈ init (paadv i)
  ∧ (σ i, ζ) = id s
  ∧ (∀j. j ≠ i → σ j ∈ (fst o id) ' init (paadv j)) } ⊆ init (opaadv i)"
  by simp
next
show "∀j. init (paadv j) ≠ {}"
  unfolding σAODV_def by simp
next
fix i s a s' σ σ'
assume "σ i = fst (id s)"
  and "σ' i = fst (id s)"
  and "(s, a, s') ∈ trans (paadv i)"
then obtain q q' where "s = (σ i, q)"
  and "s' = (σ' i, q)"
  and "((σ i, q), a, (σ' i, q')) ∈ trans (paadv i)"
  by (cases s, cases s') auto
from this(3) have "((σ, q), a, (σ', q')) ∈ trans (opaadv i)"
  by simp (rule open_seqp_action [OF aadv_wf])

with ⟨s = (σ i, q)⟩ and ⟨s' = (σ' i, q')⟩
show "((σ, snd (id s)), a, (σ', snd (id s'))) ∈ trans (opaadv i)"
  by simp
qed
then interpret op: openproc paadv opaadv id .
have [simp]: "∧i. (SOME x. x ∈ (fst o id) ' init (paadv i)) = aadv_init i"
  unfolding σAODV_def by simp
hence "∧i. openproc.initmissing paadv id i = initmissing i"
  unfolding op.initmissing_def op.someinit_def initmissing_def
  by (auto split: option.split)
thus "openproc.initmissing paadv id = initmissing" ..
qed

interpretation aadv_openproc_par_qmsg: openproc_parq paadv opaadv id qmsg
where "aadv_openproc_par_qmsg.netglobal = netglobal"
  and "aadv_openproc_par_qmsg.initmissing = initmissing"
proof -
show "openproc_parq paadv opaadv id qmsg"
  by (unfold_locales) simp
then interpret opq: openproc_parq paadv opaadv id qmsg .

have im: "∧σ. openproc.initmissing (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) σ
  = initmissing σ"
  unfolding opq.initmissing_def opq.someinit_def initmissing_def
  unfolding σAODV_def σQMSG_def by (clarsimp cong: option.case_cong)
thus "openproc.initmissing (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) = initmissing"
  by (rule ext)
have "∧P σ. openproc.netglobal (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) P σ
  = netglobal P σ"
  unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def
  unfolding σAODV_def σQMSG_def
  by (clarsimp cong: option.case_cong
    simp del: One_nat_def
    simp add: fst_initmissing_netgmap_default_aadv_init_netlift
    [symmetric, unfolded initmissing_def])
thus "openproc.netglobal (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) = netglobal"
  by auto

```



qed

lemma net\_nhop\_quality\_increases:

assumes "wf\_net\_tree n"

shows "closed (pnet ( $\lambda i$ . paodv i  $\langle\langle$  qmsg) n)  $\models$  netglobal

( $\lambda\sigma$ .  $\forall i$  dip. let nhip = the (nhop (rt ( $\sigma$  i)) dip)

in dip  $\in$  vD (rt ( $\sigma$  i))  $\cap$  vD (rt ( $\sigma$  nhip))  $\wedge$  nhip  $\neq$  dip

$\rightarrow$  (rt ( $\sigma$  i))  $\sqsubset_{\text{dip}}$  (rt ( $\sigma$  nhip)))"

(is "\_  $\models$  netglobal ( $\lambda\sigma$ .  $\forall i$ . ?inv  $\sigma$  i))"

proof -

from (wf\_net\_tree n)

have proto: "closed (pnet ( $\lambda i$ . paodv i  $\langle\langle$  qmsg) n)  $\models$  netglobal ( $\lambda\sigma$ .  $\forall i \in \text{net\_tree\_ips } n$ .  $\forall$  dip.

let nhip = the (nhop (rt ( $\sigma$  i)) dip)

in dip  $\in$  vD (rt ( $\sigma$  i))  $\cap$  vD (rt ( $\sigma$  nhip))  $\wedge$  nhip  $\neq$  dip

$\rightarrow$  (rt ( $\sigma$  i))  $\sqsubset_{\text{dip}}$  (rt ( $\sigma$  nhip)))"

by (rule aodv\_openproc\_par\_qmsg.close\_opnet [OF \_ onet\_nhop\_quality\_increases])

show ?thesis

unfolding invariant\_def opnet\_sos.opnet\_tau1

proof (rule, simp only: aodv\_openproc\_par\_qmsg.netglobalsimp

fst\_initmissing\_netgmap\_pair\_fst, rule allI)

fix  $\sigma$  i

assume sr: " $\sigma \in \text{reachable}$  (closed (pnet ( $\lambda i$ . paodv i  $\langle\langle$  qmsg) n)) TT"

hence " $\forall i \in \text{net\_tree\_ips } n$ . ?inv (fst (initmissing (netgmap fst  $\sigma$ ))) i"

by - (drule invariantD [OF proto],

simp only: aodv\_openproc\_par\_qmsg.netglobalsimp

fst\_initmissing\_netgmap\_pair\_fst)

thus "?inv (fst (initmissing (netgmap fst  $\sigma$ ))) i"

proof (cases "i  $\in$  net\_tree\_ips n")

assume "i  $\notin$  net\_tree\_ips n"

from sr have " $\sigma \in \text{reachable}$  (pnet ( $\lambda i$ . paodv i  $\langle\langle$  qmsg) n) TT" ..

hence "net\_ips  $\sigma = \text{net\_tree\_ips } n$ " ..

with (i  $\notin$  net\_tree\_ips n) have "i  $\notin$  net\_ips  $\sigma$ " by simp

hence "(fst (initmissing (netgmap fst  $\sigma$ ))) i = aodv\_init i"

by simp

thus ?thesis by simp

qed metis

qed

qed

#### 4.11.6 Loop freedom of AODV

theorem aodv\_loop\_freedom:

assumes "wf\_net\_tree n"

shows "closed (pnet ( $\lambda i$ . paodv i  $\langle\langle$  qmsg) n)  $\models$  netglobal ( $\lambda\sigma$ .  $\forall$  dip. irrefl ((rt\_graph  $\sigma$  dip)<sup>+</sup>))"

using assms by (rule aodv\_openproc\_par\_qmsg.netglobal\_weakenE

[OF net\_nhop\_quality\_increases inv\_to\_loop\_freedom])

end

# Chapter 5

## Variants A–D: All proposed modifications

This model combines the changes proposed in each of the individual variant models.

### 5.1 Predicates and functions used in the AODV model

```
theory E_Aodv_Data
imports E_All_ABCD
begin
```

#### 5.1.1 Sequence Numbers

Sequence numbers approximate the relative freshness of routing information.

```
definition inc :: "sqn  $\Rightarrow$  sqn"
  where "inc sn  $\equiv$  if sn = 0 then sn else sn + 1"
```

```
lemma less_than_inc [simp]: "x  $\leq$  inc x"
  unfolding inc_def by simp
```

```
lemma inc_minus_suc_0 [simp]:
  "inc x - Suc 0 = x"
  unfolding inc_def by simp
```

```
lemma inc_never_one' [simp, intro]: "inc x  $\neq$  Suc 0"
  unfolding inc_def by simp
```

```
lemma inc_never_one [simp, intro]: "inc x  $\neq$  1"
  by simp
```

#### 5.1.2 Modelling Routes

A route is a t-tuple,  $(dsn, dsk, flag, hops, nhop)$  where  $dsn$  is the ‘destination sequence number’,  $dsk$  is the ‘destination-sequence-number status’,  $flag$  is the route status,  $hops$  is the number of hops to the destination, and  $nhop$  is the next hop toward the destination.

```
type_synonym r = "sqn  $\times$  k  $\times$  f  $\times$  nat  $\times$  ip"
```

```
definition proj2 :: "r  $\Rightarrow$  sqn" ("π2")
  where "π2  $\equiv$   $\lambda$ (dsn, _, _, _, _). dsn"
```

```
definition proj3 :: "r  $\Rightarrow$  k" ("π3")
  where "π3  $\equiv$   $\lambda$ (_, dsk, _, _, _). dsk"
```

```
definition proj4 :: "r  $\Rightarrow$  f" ("π4")
  where "π4  $\equiv$   $\lambda$ (_, _, flag, _, _). flag"
```

```
definition proj5 :: "r  $\Rightarrow$  nat" ("π5")
  where "π5  $\equiv$   $\lambda$ (_, _, _, hops, _). hops"
```

```
definition proj6 :: "r  $\Rightarrow$  ip" ("π6")
```

where " $\pi_6 \equiv \lambda(\_, \_, \_, \_, \_). \text{nhip}$ ".

lemma *projs [simp]*:

```
" $\pi_2(\text{dsn}, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}) = \text{dsn}$ "
" $\pi_3(\text{dsn}, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}) = \text{dsk}$ "
" $\pi_4(\text{dsn}, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}) = \text{flag}$ "
" $\pi_5(\text{dsn}, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}) = \text{hops}$ "
" $\pi_6(\text{dsn}, \text{dsk}, \text{flag}, \text{hops}, \text{nhip}) = \text{nhip}$ "
by (clarsimp simp: proj2_def proj3_def proj4_def
    proj5_def proj6_def)+
```

lemma *proj3\_pred [intro]*: " $\llbracket P \text{ kno}; P \text{ unk} \rrbracket \implies P (\pi_3 \ x)$ "  
by (rule *k.induct*)

lemma *proj4\_pred [intro]*: " $\llbracket P \text{ val}; P \text{ inv} \rrbracket \implies P (\pi_4 \ x)$ "  
by (rule *f.induct*)

lemma *proj6\_pair\_snd [simp]*:

```
fixes dsn' r
shows " $\pi_6(\text{dsn}', \text{snd}(r)) = \pi_6(r)$ "
by (cases r) simp
```

### 5.1.3 Routing Tables

Routing tables map ip addresses to route entries.

type-synonym *rt* = "*ip*  $\rightarrow$  *r*"

syntax

```
" $\sigma_{\text{route}}$ " :: "rt  $\Rightarrow$  ip  $\rightarrow$  r" (" $\sigma_{\text{route}}'(\_, \_)$ ")
```

translations

```
" $\sigma_{\text{route}}(\text{rt}, \text{dip})$ " => "rt dip"
```

definition *sqn* :: "*rt*  $\Rightarrow$  *ip*  $\Rightarrow$  *sqn*"

```
where "sqn rt dip  $\equiv$  case  $\sigma_{\text{route}}(\text{rt}, \text{dip})$  of Some r  $\Rightarrow$   $\pi_2(r)$  | None  $\Rightarrow$  0"
```

definition *sqnf* :: "*rt*  $\Rightarrow$  *ip*  $\Rightarrow$  *k*"

```
where "sqnf rt dip  $\equiv$  case  $\sigma_{\text{route}}(\text{rt}, \text{dip})$  of Some r  $\Rightarrow$   $\pi_3(r)$  | None  $\Rightarrow$  unk"
```

abbreviation *flag* :: "*rt*  $\Rightarrow$  *ip*  $\rightarrow$  *f*"

```
where "flag rt dip  $\equiv$  map_option  $\pi_4$  ( $\sigma_{\text{route}}(\text{rt}, \text{dip})$ )"
```

abbreviation *dhops* :: "*rt*  $\Rightarrow$  *ip*  $\rightarrow$  nat"

```
where "dhops rt dip  $\equiv$  map_option  $\pi_5$  ( $\sigma_{\text{route}}(\text{rt}, \text{dip})$ )"
```

abbreviation *nhop* :: "*rt*  $\Rightarrow$  *ip*  $\rightarrow$  *ip*"

```
where "nhop rt dip  $\equiv$  map_option  $\pi_6$  ( $\sigma_{\text{route}}(\text{rt}, \text{dip})$ )"
```

definition *vD* :: "*rt*  $\Rightarrow$  *ip* set"

```
where "vD rt  $\equiv$  {dip. flag rt dip = Some val}"
```

definition *iD* :: "*rt*  $\Rightarrow$  *ip* set"

```
where "iD rt  $\equiv$  {dip. flag rt dip = Some inv}"
```

definition *kD* :: "*rt*  $\Rightarrow$  *ip* set"

```
where "kD rt  $\equiv$  {dip. rt dip  $\neq$  None}"
```

lemma *kD\_is\_vD\_and\_iD*: "*kD* *rt* = *vD* *rt*  $\cup$  *iD* *rt*"

```
unfolding kD_def vD_def iD_def by auto
```

lemma *vD\_iD\_gives\_kD [simp]*:

```
" $\bigwedge ip \text{ rt}. ip \in vD \text{ rt} \implies ip \in kD \text{ rt}$ "
```

```
" $\bigwedge ip \text{ rt}. ip \in iD \text{ rt} \implies ip \in kD \text{ rt}$ "
```

```
unfolding kD_is_vD_and_iD by simp_all
```

```

lemma kD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ kD rt"
  shows "∃ dsn dsk flag hops nhip.
    σroute(rt, dip) = Some (dsn, dsk, flag, hops, nhip)"
  using assms unfolding kD_def by simp

lemma kD_None [dest]:
  fixes dip rt
  assumes "dip ∉ kD rt"
  shows "σroute(rt, dip) = None"
  using assms unfolding kD_def
  by (metis (mono_tags) mem_Collect_eq)

lemma vD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ vD rt"
  shows "∃ dsn dsk hops nhip.
    σroute(rt, dip) = Some (dsn, dsk, val, hops, nhip)"
  using assms unfolding vD_def by simp

lemma vD_empty [simp]: "vD Map.empty = {}"
  unfolding vD_def by simp

lemma iD_Some [dest]:
  fixes dip rt
  assumes "dip ∈ iD rt"
  shows "∃ dsn dsk hops nhip.
    σroute(rt, dip) = Some (dsn, dsk, inv, hops, nhip)"
  using assms unfolding iD_def by simp

lemma val_is_vD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "the (flag rt ip) = val"
  shows "ip ∈ vD(rt)"
  using assms unfolding vD_def by auto

lemma inv_is_iD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "the (flag rt ip) = inv"
  shows "ip ∈ iD(rt)"
  using assms unfolding iD_def by auto

lemma iD_flag_is_inv [elim, simp]:
  fixes ip rt
  assumes "ip ∈ iD(rt)"
  shows "the (flag rt ip) = inv"
  proof -
    from ⟨ip ∈ iD(rt)⟩ have "ip ∈ kD(rt)" by auto
    with assms show ?thesis unfolding iD_def by auto
  qed

lemma kD_but_not_vD_is_iD [elim]:
  fixes ip rt
  assumes "ip ∈ kD(rt)"
  and "ip ∉ vD(rt)"
  shows "ip ∈ iD(rt)"
  proof -
    from ⟨ip ∈ kD(rt)⟩ obtain dsn dsk f hops nhop
      where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop)"
      by (metis kD_Some)
  
```

```

from ⟨ip∉vD(rt)⟩ have "f ≠ val"
proof (rule contrapos_nn)
  assume "f = val"
  with rtip have "the (flag rt ip) = val" by simp
  with ⟨ip∈kD(rt)⟩ show "ip∈vD(rt)" ..
qed
with rtip have "the (flag rt ip) = inv" by simp
with ⟨ip∈kD(rt)⟩ show "ip∈iD(rt)" ..
qed

lemma vD_or_iD [elim]:
  fixes ip rt
  assumes "ip∈kD(rt)"
    and "ip∈vD(rt) ⇒ P rt ip"
    and "ip∈iD(rt) ⇒ P rt ip"
  shows "P rt ip"
proof -
  from ⟨ip∈kD(rt)⟩ have "ip∈vD(rt) ∪ iD(rt)"
  by (simp add: kD_is_vD_and_iD)
  thus ?thesis by (auto elim: assms(2-3))
qed

lemma proj5_eq_dhops: "∧dip rt. dip∈kD(rt) ⇒ π5(the (rt dip)) = the (dhops rt dip)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj4_eq_flag: "∧dip rt. dip∈kD(rt) ⇒ π4(the (rt dip)) = the (flag rt dip)"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj2_eq_sqn: "∧dip rt. dip∈kD(rt) ⇒ π2(the (rt dip)) = sqn rt dip"
unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 [simp]:
  "∧ip rt. ip∈kD(rt) ⇒ sqnf rt ip = π3(the (rt ip))"
unfolding sqnf_def by auto

lemma vD_flag_val [simp]:
  "∧dip rt. dip ∈ vD (rt) ⇒ the (flag rt dip) = val"
unfolding vD_def by clarsimp

lemma kD_update [simp]:
  "∧rt nip v. kD (rt(nip ↦ v)) = insert nip (kD rt)"
unfolding kD_def by auto

lemma kD_empty [simp]: "kD Map.empty = {}"
unfolding kD_def by simp

lemma ip_equal_or_known [elim]:
  fixes rt ip ip'
  assumes "ip = ip' ∨ ip∈kD(rt)"
    and "ip = ip' ⇒ P rt ip ip'"
    and "[| ip ≠ ip'; ip∈kD(rt) |] ⇒ P rt ip ip'"
  shows "P rt ip ip'"
using assms by auto

```

#### 5.1.4 Updating Routing Tables

Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.

##### Updating route entries

```

lemma in_kD_case [simp]:
  fixes dip rt
  assumes "dip ∈ kD(rt)"

```

```

  shows "(case rt dip of None  $\Rightarrow$  en | Some r  $\Rightarrow$  es r) = es (the (rt dip))"
using assms [THEN kD_Some] by auto

lemma not_in_kD_case [simp]:
  fixes dip rt
  assumes "dip  $\notin$  kD(rt)"
  shows "(case rt dip of None  $\Rightarrow$  en | Some r  $\Rightarrow$  es r) = en"
using assms [THEN kD_None] by auto

lemma rt_Some_sqn [dest]:
  fixes rt and ip dsn dsk flag hops nhip
  assumes "rt ip = Some (dsn, dsk, flag, hops, nhip)"
  shows "sqn rt ip = dsn"
unfolding sqn_def using assms by simp

lemma not_kD_sqn [simp]:
  fixes dip rt
  assumes "dip  $\notin$  kD(rt)"
  shows "sqn rt dip = 0"
using assms unfolding sqn_def
by simp

definition update_arg_wf :: "r  $\Rightarrow$  bool"
where "update_arg_wf r  $\equiv$   $\pi_4(r) = \text{val} \wedge$ 
      ( $\pi_2(r) = 0$ ) = ( $\pi_3(r) = \text{unk}$ )  $\wedge$ 
      ( $\pi_3(r) = \text{unk} \longrightarrow \pi_5(r) = 1$ )"

lemma update_arg_wf_gives_cases:
  " $\bigwedge r$ . update_arg_wf r  $\implies$  ( $\pi_2(r) = 0$ ) = ( $\pi_3(r) = \text{unk}$ )"
unfolding update_arg_wf_def by simp

lemma update_arg_wf_tuples [simp]:
  " $\bigwedge nhip$ . update_arg_wf (0, unk, val, Suc 0, nhip)"
  " $\bigwedge n$  hops nhip. update_arg_wf (Suc n, kno, val, hops, nhip)"
unfolding update_arg_wf_def by auto

lemma update_arg_wf_tuples' [elim]:
  " $\bigwedge n$  hops nhip. Suc 0  $\leq n \implies$  update_arg_wf (n, kno, val, hops, nhip)"
unfolding update_arg_wf_def by auto

lemma wf_r_cases [intro]:
  fixes P r
  assumes "update_arg_wf r"
  and c1: " $\bigwedge nhip$ . P (0, unk, val, Suc 0, nhip)"
  and c2: " $\bigwedge dsn$  hops nhip. dsn > 0  $\implies$  P (dsn, kno, val, hops, nhip)"
  shows "P r"
proof -
  obtain dsn dsk flag hops nhip
  where *: "r = (dsn, dsk, flag, hops, nhip)" by (cases r)
  with (update_arg_wf r) have wf1: "flag = val"
  and wf2: "(dsn = 0) = (dsk = unk)"
  and wf3: "dsk = unk  $\longrightarrow$  (hops = 1)"
  unfolding update_arg_wf_def by auto
  have "P (dsn, dsk, flag, hops, nhip)"
  proof (cases dsk)
    assume "dsk = unk"
    moreover with wf2 wf3 have "dsn = 0" and "hops = Suc 0" by auto
    ultimately show ?thesis using (flag = val) by simp (rule c1)
  next
    assume "dsk = kno"
    moreover with wf2 have "dsn > 0" by simp
    ultimately show ?thesis using (flag = val) by simp (rule c2)
  qed
  with * show "P r" by simp

```

qed

definition update :: "rt  $\Rightarrow$  ip  $\Rightarrow$  r  $\Rightarrow$  rt"

where

```
"update rt ip r  $\equiv$ 
  case  $\sigma_{route}(rt, ip)$  of
    None  $\Rightarrow$  rt (ip  $\mapsto$  r)
  | Some s  $\Rightarrow$ 
    if  $\pi_2(s) < \pi_2(r)$  then rt (ip  $\mapsto$  r)
    else if  $\pi_2(s) = \pi_2(r) \wedge (\pi_5(s) > \pi_5(r) \vee \pi_4(s) = inv)$ 
      then rt (ip  $\mapsto$  r)
    else if  $\pi_3(r) = unk$ 
      then rt (ip  $\mapsto$  ( $\pi_2(s)$ , snd (r)))
    else rt (ip  $\mapsto$  s)"
```

lemma update\_simps [simp]:

fixes r s nrt nr' ns rt ip

defines "s  $\equiv$  the  $\sigma_{route}(rt, ip)$ "

and "nr'  $\equiv$  ( $\pi_2(s)$ ,  $\pi_3(r)$ ,  $\pi_4(r)$ ,  $\pi_5(r)$ ,  $\pi_6(r)$ )"

shows

```
"[ip  $\notin$  kD(rt)]  $\implies$  update rt ip r = rt (ip  $\mapsto$  r)"
"[ip  $\in$  kD(rt); sqn rt ip <  $\pi_2(r)$ ]  $\implies$  update rt ip r = rt (ip  $\mapsto$  r)"
"[ip  $\in$  kD(rt); sqn rt ip =  $\pi_2(r)$ ;
  the (dhops rt ip) >  $\pi_5(r)$ ]  $\implies$  update rt ip r = rt (ip  $\mapsto$  r)"
"[ip  $\in$  kD(rt); sqn rt ip =  $\pi_2(r)$ ;
  flag rt ip = Some inv]  $\implies$  update rt ip r = rt (ip  $\mapsto$  r)"
"[ip  $\in$  kD(rt);  $\pi_3(r) = unk$ ; ( $\pi_2(r) = 0$ ) = ( $\pi_3(r) = unk$ )]  $\implies$  update rt ip r = rt (ip  $\mapsto$  nr)"
"[ip  $\in$  kD(rt); sqn rt ip  $\geq$   $\pi_2(r)$ ;  $\pi_3(r) = kno$ ;
  sqn rt ip =  $\pi_2(r)$ ]  $\implies$  the (dhops rt ip)  $\leq$   $\pi_5(r) \wedge$  the (flag rt ip) = val ]
 $\implies$  update rt ip r = rt (ip  $\mapsto$  s)"
```

proof -

assume "ip  $\notin$  kD(rt)"

hence " $\sigma_{route}(rt, ip) = None$ " ..

thus "update rt ip r = rt (ip  $\mapsto$  r)"

unfolding update\_def by simp

next

assume "ip  $\in$  kD(rt)"

and "sqn rt ip <  $\pi_2(r)$ "

from this(1) obtain dsn dsk fl hops nhip

where "rt ip = Some (dsn, dsk, fl, hops, nhip)"

by (metis kD\_Some)

with (sqn rt ip <  $\pi_2(r)$ ) show "update rt ip r = rt (ip  $\mapsto$  r)"

unfolding update\_def s\_def by auto

next

assume "ip  $\in$  kD(rt)"

and "sqn rt ip =  $\pi_2(r)$ "

and "the (dhops rt ip) >  $\pi_5(r)$ "

from this(1) obtain dsn dsk fl hops nhip

where "rt ip = Some (dsn, dsk, fl, hops, nhip)"

by (metis kD\_Some)

with (sqn rt ip =  $\pi_2(r)$ ) and (the (dhops rt ip) >  $\pi_5(r)$ )

show "update rt ip r = rt (ip  $\mapsto$  r)"

unfolding update\_def s\_def by auto

next

assume "ip  $\in$  kD(rt)"

and "sqn rt ip =  $\pi_2(r)$ "

and "flag rt ip = Some inv"

from this(1) obtain dsn dsk fl hops nhip

where "rt ip = Some (dsn, dsk, fl, hops, nhip)"

by (metis kD\_Some)

with (sqn rt ip =  $\pi_2(r)$ ) and (flag rt ip = Some inv)

show "update rt ip r = rt (ip  $\mapsto$  r)"

unfolding update\_def s\_def by auto

next

```

assume "ip ∈ kD(rt)"
  and "π3(r) = unk"
  and "(π2(r) = 0) = (π3(r) = unk)"
from this(1) obtain dsn dsk fl hops nhip
  where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
  by (metis kD_Some)
with ⟨(π2(r) = 0) = (π3(r) = unk)⟩ and ⟨π3(r) = unk⟩
  show "update rt ip r = rt (ip ↦ nr)"
  unfolding update_def nr'_def s_def
  by (cases r) simp
next
assume "ip ∈ kD(rt)"
  and otherassms: "sqn rt ip ≥ π2(r)"
  "π3(r) = kno"
  "sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val"
from this(1) obtain dsn dsk fl hops nhip
  where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
  by (metis kD_Some)
with otherassms show "update rt ip r = rt (ip ↦ s)"
  unfolding update_def s_def by auto
qed

```

lemma update\_cases [elim]:

```

assumes "(π2(r) = 0) = (π3(r) = unk)"
  and c1: "[ip ∉ kD(rt)] ⇒ P (rt (ip ↦ r))"

  and c2: "[ip ∈ kD(rt); sqn rt ip < π2(r)]
    ⇒ P (rt (ip ↦ r))"
  and c3: "[ip ∈ kD(rt); sqn rt ip = π2(r); the (dhops rt ip) > π5(r)]
    ⇒ P (rt (ip ↦ r))"
  and c4: "[ip ∈ kD(rt); sqn rt ip = π2(r); the (flag rt ip) = inv]
    ⇒ P (rt (ip ↦ r))"
  and c5: "[ip ∈ kD(rt); π3(r) = unk]
    ⇒ P (rt (ip ↦ (π2(the σroute(rt, ip)), π3(r),
      π4(r), π5(r), π6(r))))"
  and c6: "[ip ∈ kD(rt); sqn rt ip ≥ π2(r); π3(r) = kno;
    sqn rt ip = π2(r) ⇒ the (dhops rt ip) ≤ π5(r) ∧ the (flag rt ip) = val]
    ⇒ P (rt (ip ↦ the σroute(rt, ip)))"

```

shows "(P (update rt ip r))"

proof (cases "ip ∈ kD(rt)")

```

  assume "ip ∉ kD(rt)"
  with c1 show ?thesis
  by simp

```

next

```

  assume "ip ∈ kD(rt)"
  moreover then obtain dsn dsk fl hops nhip
    where rteq: "rt ip = Some (dsn, dsk, fl, hops, nhip)"
    by (metis kD_Some)
  moreover obtain dsn' dsk' fl' hops' nhip'
    where req: "r = (dsn', dsk', fl', hops', nhip' )"
    by (cases r) metis
  ultimately show ?thesis
  using ⟨(π2(r) = 0) = (π3(r) = unk)⟩
    c2 [OF ⟨ip ∈ kD(rt)⟩]
    c3 [OF ⟨ip ∈ kD(rt)⟩]
    c4 [OF ⟨ip ∈ kD(rt)⟩]
    c5 [OF ⟨ip ∈ kD(rt)⟩]
    c6 [OF ⟨ip ∈ kD(rt)⟩]
  unfolding update_def sqn_def by auto
qed

```

lemma update\_cases\_kD:

```

assumes "(π2(r) = 0) = (π3(r) = unk)"
  and "ip ∈ kD(rt)"

```



```

and c2: "sqn rt ip <  $\pi_2(r) \implies P (rt (ip \mapsto r))"$ 
and c3: "[[sqn rt ip =  $\pi_2(r)$ ; the (dhops rt ip) >  $\pi_5(r)$ ]]
 $\implies P (rt (ip \mapsto r))"$ 
and c4: "[[sqn rt ip =  $\pi_2(r)$ ; the (flag rt ip) = inv]]
 $\implies P (rt (ip \mapsto r))"$ 
and c5: " $\pi_3(r) = \text{unk} \implies P (rt (ip \mapsto (\pi_2(\text{the } \sigma_{\text{route}}(rt, ip)), \pi_3(r),$ 
 $\pi_4(r), \pi_5(r), \pi_6(r))))"$ 
and c6: "[[sqn rt ip  $\geq \pi_2(r)$ ;  $\pi_3(r) = \text{kno}$ ;
sqn rt ip =  $\pi_2(r) \implies \text{the (dhops rt ip)} \leq \pi_5(r) \wedge \text{the (flag rt ip)} = \text{val}$ ]]
 $\implies P (rt (ip \mapsto \text{the } \sigma_{\text{route}}(rt, ip)))"$ 
shows "(P (update rt ip r))"
using assms(1) proof (rule update_cases)
assume "sqn rt ip <  $\pi_2(r)$ "
thus "P (rt(ip  $\mapsto r$ ))" by (rule c2)
next
assume "sqn rt ip =  $\pi_2(r)$ "
and "the (dhops rt ip) >  $\pi_5(r)$ "
thus "P (rt(ip  $\mapsto r$ ))"
by (rule c3)
next
assume "sqn rt ip =  $\pi_2(r)$ "
and "the (flag rt ip) = inv"
thus "P (rt(ip  $\mapsto r$ ))"
by (rule c4)
next
assume " $\pi_3(r) = \text{unk}$ "
thus "P (rt (ip  $\mapsto (\pi_2(\text{the } \sigma_{\text{route}}(rt, ip)), \pi_3(r), \pi_4(r), \pi_5(r), \pi_6(r))))"$ 
by (rule c5)
next
assume "sqn rt ip  $\geq \pi_2(r)$ "
and " $\pi_3(r) = \text{kno}$ "
and "sqn rt ip =  $\pi_2(r) \implies \text{the (dhops rt ip)} \leq \pi_5(r) \wedge \text{the (flag rt ip)} = \text{val}$ "
thus "P (rt (ip  $\mapsto \text{the (rt ip)}))"$ 
by (rule c6)
qed (simp add: ⟨ip  $\in kD(rt)$ ⟩)

```

```

lemma in_kD_after_update [simp]:
fixes rt nip dsn dsk flag hops nhip pre
shows "kD (update rt nip (dsn, dsk, flag, hops, nhip)) = insert nip (kD rt)"
unfolding update_def
by (cases "rt nip") auto

```

```

lemma nhop_of_update [simp]:
fixes rt dip dsn dsk flag hops nhip
assumes "rt  $\neq$  update rt dip (dsn, dsk, flag, hops, nhip)"
shows "the (nhop (update rt dip (dsn, dsk, flag, hops, nhip)) dip) = nhip"
proof -
from assms
have update_neq: " $\bigwedge v. rt \text{ dip} = \text{Some } v \implies$ 
update rt dip (dsn, dsk, flag, hops, nhip)
 $\neq$  rt(dip  $\mapsto \text{the (rt dip)})"$ 
by auto
show ?thesis
proof (cases "rt dip = None")
assume "rt dip = None"
thus "?thesis" unfolding update_def by clarsimp
next
assume "rt dip  $\neq$  None"
then obtain v where "rt dip = Some v" by (metis not_None_eq)
with update_neq [OF this] show ?thesis
unfolding update_def by auto
qed
qed

```

```

lemma sqn_if_updated:
  fixes rip v rt ip
  shows "sqn ( $\lambda x. \text{if } x = \text{rip} \text{ then Some } v \text{ else } \text{rt } x$ ) ip
        = (if ip = rip then  $\pi_2(v)$  else sqn rt ip)"
  unfolding sqn_def by simp

lemma update_sqn [simp]:
  fixes rt dip rip dsn dsk hops nhip
  assumes "(dsn = 0) = (dsk = unk)"
  shows "sqn rt dip  $\leq$  sqn (update rt rip (dsn, dsk, val, hops, nhip)) dip"
  proof (rule update_cases)
    show "( $\pi_2$  (dsn, dsk, val, hops, nhip) = 0) = ( $\pi_3$  (dsn, dsk, val, hops, nhip) = unk)"
    by simp (rule assms)
  qed (clarsimp simp: sqn_if_updated sqn_def)+

lemma sqn_update_bigger [simp]:
  fixes rt ip ip' dsn dsk flag hops nhip
  assumes "1  $\leq$  hops"
  shows "sqn rt ip  $\leq$  sqn (update rt ip' (dsn, dsk, flag, hops, nhip)) ip"
  using assms unfolding update_def sqn_def
  by (clarsimp split: option.split) auto

lemma dhops_update [intro]:
  fixes rt dsn dsk flag hops ip rip nhip
  assumes ex: " $\forall ip \in kD \text{ rt. the } (dhops \text{ rt } ip) \geq 1$ "
    and ip: "(ip = rip  $\wedge$  Suc 0  $\leq$  hops)  $\vee$  (ip  $\neq$  rip  $\wedge$  ip  $\in kD \text{ rt})$ "
  shows "Suc 0  $\leq$  the (dhops (update rt rip (dsn, dsk, flag, hops, nhip)) ip)"
  using ip proof
    assume "ip = rip  $\wedge$  Suc 0  $\leq$  hops" thus ?thesis
      unfolding update_def using ex
      by (cases "rip  $\in kD \text{ rt}")$  (drule(1) bspec, auto)
  next
    assume "ip  $\neq$  rip  $\wedge$  ip  $\in kD \text{ rt}"$  thus ?thesis
      using ex unfolding update_def
      by (cases "rip  $\in kD \text{ rt}")$  auto
  qed

lemma update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip  $\neq$  dip"
  shows "(update rt dip (dsn, dsk, flag, hops, nhip)) ip = rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma nhop_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip  $\neq$  dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nhip)) ip = nhop rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma dhops_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip  $\neq$  dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nhip)) ip = dhops rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma sqn_update_same [simp]:
  " $\bigwedge \text{rt ip dsn dsk flag hops nhip. sqn } (\text{rt}(ip \mapsto v)) \text{ ip} = \pi_2(v)$ "
  unfolding sqn_def by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nhip

```

```

assumes "rt ≠ update rt dip (osn, kno, val, hops, nhip)"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nhip)) dip) = hops"
using assms unfolding update_def
by (clarsimp split: option.split_asm option.split_split_if_asm) auto

lemma nhop_update_unk_val [simp]:
  "∧rt dip ip dsn hops.
  the (nhop (update rt dip (dsn, unk, val, hops, ip)) dip) = ip"
  unfolding update_def by (clarsimp split: option.split)

lemma nhop_update_changed [simp]:
  fixes rt dip dsn dsk flg hops sip
  assumes "update rt dip (dsn, dsk, flg, hops, sip) ≠ rt"
  shows "the (nhop (update rt dip (dsn, dsk, flg, hops, sip)) dip) = sip"
  using assms unfolding update_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma update_rt_split_asm:
  "∧rt ip dsn dsk flag hops sip.
  P (update rt ip (dsn, dsk, flag, hops, sip))
  =
  (¬(rt = update rt ip (dsn, dsk, flag, hops, sip)) ∧ ¬P rt
  ∨ rt ≠ update rt ip (dsn, dsk, flag, hops, sip)
  ∧ ¬P (update rt ip (dsn, dsk, flag, hops, sip)))"
  by auto

lemma sqn_update [simp]: "∧rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip)
  ⇒ sqn (update rt dip (dsn, kno, flg, hops, sip)) dip = dsn"
  unfolding update_def by (clarsimp split: option.split_split_if_asm) auto

lemma sqnf_update [simp]: "∧rt dip dsn dsk flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip)
  ⇒ sqnf (update rt dip (dsn, dsk, flg, hops, sip)) dip = dsk"
  unfolding update_def sqnf_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma update_kno_dsn_greater_zero:
  "∧rt dip ip dsn hops. 1 ≤ dsn ⇒ 1 ≤ (sqn (update rt dip (dsn, kno, val, hops, ip)) dip)"
  unfolding update_def
  by (clarsimp split: option.splits)

lemma proj3_update [simp]: "∧rt dip dsn dsk flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip)
  ⇒ π3(the (update rt dip (dsn, dsk, flg, hops, sip) dip)) = dsk"
  unfolding update_def sqnf_def
  by (clarsimp split: option.splits split_if_asm) auto

lemma nhop_update_changed_kno_val [simp]: "∧rt ip dsn dsk hops nhip.
  rt ≠ update rt ip (dsn, kno, val, hops, nhip)
  ⇒ the (nhop (update rt ip (dsn, kno, val, hops, nhip)) ip) = nhip"
  using assms unfolding update_def
  by (clarsimp split: option.split_asm option.split_split_if_asm) auto

lemma flag_update [simp]: "∧rt dip dsn flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip)
  ⇒ the (flag (update rt dip (dsn, kno, flg, hops, sip)) dip) = flg"
  unfolding update_def
  by (clarsimp split: option.split_split_if_asm) auto

lemma the_flag_Some [dest!]:
  fixes ip rt
  assumes "the (flag rt ip) = x"
  and "ip ∈ kD rt"

```

```

shows "flag rt ip = Some x"
using assms by auto

```

```

lemma kD_update_unchanged [dest]:
  fixes rt dip dsn dsk flag hops nhip
  assumes "rt = update rt dip (dsn, dsk, flag, hops, nhip)"
  shows "dip ∈ kD(rt)"
proof -
  have "dip ∈ kD(update rt dip (dsn, dsk, flag, hops, nhip))" by simp
  with assms show ?thesis by simp
qed

```

```

lemma nhop_update [simp]: "∧rt dip dsn dsk flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip)
  ⇒ the (nhop (update rt dip (dsn, dsk, flg, hops, sip)) dip) = sip"
unfolding update_def sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

```

```

lemma sqn_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip ≠ dip"
  shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip)) ip = sqn rt ip"
using assms unfolding update_def sqn_def
by (clarsimp split: option.splits) auto

```

```

lemma sqnf_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip ≠ dip"
  shows "sqnf (update rt dip (dsn, dsk, flag, hops, nhip)) ip = sqnf rt ip"
using assms unfolding update_def sqnf_def
by (clarsimp split: option.splits) auto

```

```

lemma vD_update_val [dest]:
  "∧dip rt dip' dsn dsk hops nhip.
  dip ∈ vD(update rt dip' (dsn, dsk, val, hops, nhip)) ⇒ (dip ∈ vD(rt) ∨ dip=dip')"
unfolding update_def vD_def by (clarsimp split: option.split_asm split_if_asm)

```

## Invalidating route entries

```

definition invalidate :: "rt ⇒ (ip → sqn) ⇒ rt"
where "invalidate rt dests ≡
  λip. case (rt ip, dests ip) of
    (None, _) ⇒ None
  | (Some s, None) ⇒ Some s
  | (Some (_, dsk, _, hops, nhip), Some rsn) ⇒
    Some (rsn, dsk, inv, hops, nhip)"

```

```

lemma proj3_invalidate [simp]:
  "∧dip. π3(the ((invalidate rt dests) dip)) = π3(the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)

```

```

lemma proj5_invalidate [simp]:
  "∧dip. π5(the ((invalidate rt dests) dip)) = π5(the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)

```

```

lemma proj6_invalidate [simp]:
  "∧dip. π6(the ((invalidate rt dests) dip)) = π6(the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)

```

### 5.1.5 Route Requests

```

lemma invalidate_kD_inv [simp]:
  "∧rt dests. kD (invalidate rt dests) = kD rt"
unfolding invalidate_def kD_def
by (simp split: option.split)

```

```

lemma invalidate_sqn:
  fixes rt dip dests
  assumes "∀rsn. dests dip = Some rsn → sqn rt dip ≤ rsn"
  shows "sqn rt dip ≤ sqn (invalidate rt dests) dip"
  proof (cases "dip ∈ kD(rt)")
    assume "¬ dip ∈ kD(rt)"
    hence "dip ∈ kD(rt)" by simp
    then obtain dsn dsk flag hops nhip where "rt dip = Some (dsn, dsk, flag, hops, nhip)"
      by (metis kD_Some)
    with assms show "sqn rt dip ≤ sqn (invalidate rt dests) dip"
      by (cases "dests dip") (auto simp add: invalidate_def sqn_def)
  qed simp

lemma sqn_invalidate_in_dests [simp]:
  fixes dests ipa rsn rt
  assumes "dests ipa = Some rsn"
    and "ipa ∈ kD(rt)"
  shows "sqn (invalidate rt dests) ipa = rsn"
  unfolding invalidate_def sqn_def
  using assms(1) assms(2) [THEN kD_Some]
  by clarsimp

lemma dhops_invalidate [simp]:
  "∧dip. the (dhops (invalidate rt dests) dip) = the (dhops rt dip)"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma sqnf_invalidate [simp]:
  "∧dip. sqnf (invalidate (rt ξ) (dests ξ)) dip = sqnf (rt ξ) dip"
  unfolding sqnf_def invalidate_def by (clarsimp split: option.split)

lemma nhop_invalidate [simp]:
  "∧dip. the (nhop (invalidate (rt ξ) (dests ξ)) dip) = the (nhop (rt ξ) dip)"
  unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
  fixes rt dests dip
  assumes "dip ∉ dom(dests)"
  shows "invalidate rt dests dip = rt dip"
  using assms unfolding invalidate_def
  by (clarsimp split: option.split_asm)

lemma invalidate_none [simp]:
  fixes rt dests dip
  assumes "dip ∉ kD(rt)"
  shows "invalidate rt dests dip = None"
  using assms unfolding invalidate_def by clarsimp

lemma vD_invalidate_vD_not_dests:
  "∧dip rt dests. dip ∈ vD (invalidate rt dests) ⇒ dip ∈ vD(rt) ∧ dests dip = None"
  unfolding invalidate_def vD_def
  by (clarsimp split: option.split_asm)

lemma sqn_invalidate_not_in_dests [simp]:
  fixes dests dip rt
  assumes "dip ∉ dom(dests)"
  shows "sqn (invalidate rt dests) dip = sqn rt dip"
  using assms unfolding sqn_def by simp

lemma invalidate_changes:
  fixes rt dests dip dsn dsk flag hops nhip pre
  assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip)"
  shows " dsn = (case dests dip of None ⇒ π2(the (rt dip)) | Some rsn ⇒ rsn)
    ∧ dsk = π3(the (rt dip))"

```

```

    ∧ flag = (if dests dip = None then  $\pi_4$ (the (rt dip)) else inv)
    ∧ hops =  $\pi_5$ (the (rt dip))
    ∧ nhip =  $\pi_6$ (the (rt dip))"
using assms unfolding invalidate_def
by (cases "rt dip", clarsimp, cases "dests dip") auto

```

```

lemma proj3_inv: "∧dip rt dests. dip ∈ kD (rt)
  ⇒  $\pi_3$ (the (invalidate rt dests dip)) =  $\pi_3$ (the (rt dip))"
by (clarsimp simp: invalidate_def kD_def split: option.split)

```

```

lemma dests_iD_invalidate [simp]:
  assumes "dests ip = Some rsn"
    and "ip ∈ kD(rt)"
  shows "ip ∈ iD(invalidate rt dests)"
using assms(1) assms(2) [THEN kD_Some] unfolding invalidate_def iD_def
by (clarsimp split: option.split)

```

### 5.1.6 Queued Packets

Functions for sending data packets.

```

type_synonym store = "ip → (p × data list)"

```

```

definition sigma_queue :: "store ⇒ ip ⇒ data list" ("σqueue'(_, _)'")
  where "σqueue(store, dip) ≡ case store dip of None ⇒ [] | Some (p, q) ⇒ q"

```

```

definition qD :: "store ⇒ ip set"
  where "qD ≡ dom"

```

```

definition add :: "data ⇒ ip ⇒ store ⇒ store"
  where "add d dip store ≡ case store dip of
    None ⇒ store (dip ↦ (req, [d]))
    | Some (p, q) ⇒ store (dip ↦ (p, q @ [d]))"

```

```

lemma qD_add [simp]:
  fixes d dip store
  shows "qD(add d dip store) = insert dip (qD store)"
  unfolding add_def Let_def qD_def
  by (clarsimp split: option.split)

```

```

definition drop :: "ip ⇒ store → store"
  where "drop dip store ≡
  map_option (λ(p, q). if tl q = [] then store (dip := None)
    else store (dip ↦ (p, tl q))) (store dip)"

```

```

definition sigma_p_flag :: "store ⇒ ip → p" ("σp-flag'(_, _)'")
  where "σp-flag(store, dip) ≡ map_option fst (store dip)"

```

```

definition unsetRRF :: "store ⇒ ip ⇒ store"
  where "unsetRRF store dip ≡ case store dip of
    None ⇒ store
    | Some (p, q) ⇒ store (dip ↦ (noreq, q))"

```

```

definition setRRF :: "store ⇒ (ip → sqn) ⇒ store"
  where "setRRF store dests ≡ λdip. if dests dip = None then store dip
    else map_option (λ(_, q). (req, q)) (store dip)"

```

### 5.1.7 Comparison with the original technical report

The major differences with the AODV technical report of Fehnker et al are:

1. *nhop* is partial, thus a ‘the’ is needed, similarly for *dhops* and *addpreRT*.
2. *prec*s is partial.

3.  $\sigma_{p\text{-flag}}(\text{store}, \text{dip})$  is partial.
4. The routing table ( $\text{rt}$ ) is modelled as a map ( $\text{ip} \Rightarrow \text{r option}$ ) rather than a set of 7-tuples, likewise, the  $\text{r}$  is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address ( $\text{dip}$ ) is taken from the argument to the function, rather than a part of the result. Well-definedness then follows from the structure of the type and more related facts are available automatically, rather than having to be acquired through tedious proofs.
5. Similar remarks hold for the  $\text{dests}$  mapping passed to  $\text{invalidate}$ , and  $\text{store}$ .

end

## 5.2 AODV protocol messages

```

theory E_Aodv_Message
imports E_All_ABCD
begin

datatype msg =
  Rreq nat ip sqn k ip sqn ip bool
  | Rrep nat ip sqn ip ip
  | Rerr "ip  $\rightarrow$  sqn" ip
  | Newpkt data ip
  | Pkt data ip ip

instantiation msg :: msg
begin
  definition newpkt_def [simp]: "newpkt  $\equiv$   $\lambda(d, \text{dip}). \text{Newpkt } d \text{ dip}"$ 
  definition eq_newpkt_def: "eq_newpkt  $m \equiv$  case  $m$  of  $\text{Newpkt } d \text{ dip} \Rightarrow \text{True} \mid \_ \Rightarrow \text{False}"$ 

  instance by intro_classes (simp add: eq_newpkt_def)
end

```

The  $\text{msg}$  type models the different messages used within AODV. The instantiation as a  $\text{msg}$  is a technicality due to the special treatment of  $\text{newpkt}$  messages in the AWN SOS rules. This use of classes allows a clean separation of the AWN-specific definitions and these AODV-specific definitions.

```

definition rreq :: "nat  $\times$  ip  $\times$  sqn  $\times$  k  $\times$  ip  $\times$  sqn  $\times$  ip  $\times$  bool  $\Rightarrow$  msg"
  where "rreq  $\equiv$   $\lambda(\text{hops}, \text{dip}, \text{dsn}, \text{dsk}, \text{oip}, \text{osn}, \text{sip}, \text{handled}).$ 
         Rreq hops dip dsn dsk oip osn sip handled"

```

```

lemma rreq_simp [simp]:
  "rreq(hops, dip, dsn, dsk, oip, osn, sip, handled) = Rreq hops dip dsn dsk oip osn sip handled"
  unfolding rreq_def by simp

```

```

definition rrep :: "nat  $\times$  ip  $\times$  sqn  $\times$  ip  $\times$  ip  $\Rightarrow$  msg"
  where "rrep  $\equiv$   $\lambda(\text{hops}, \text{dip}, \text{dsn}, \text{oip}, \text{sip}). \text{Rrep } \text{hops } \text{dip } \text{dsn } \text{oip } \text{sip}"$ 

```

```

lemma rrep_simp [simp]:
  "rrep(hops, dip, dsn, oip, sip) = Rrep hops dip dsn oip sip"
  unfolding rrep_def by simp

```

```

definition rerr :: "(ip  $\rightarrow$  sqn)  $\times$  ip  $\Rightarrow$  msg"
  where "rerr  $\equiv$   $\lambda(\text{dests}, \text{sip}). \text{Rerr } \text{dests } \text{sip}"$ 

```

```

lemma rerr_simp [simp]:
  "rerr(dests, sip) = Rerr dests sip"
  unfolding rerr_def by simp

```

```

lemma not_eq_newpkt_rreq [simp]: " $\neg \text{eq\_newpkt } (\text{Rreq } \text{hops } \text{dip } \text{dsn } \text{dsk } \text{oip } \text{osn } \text{sip } \text{handled})"$ 
  unfolding eq_newpkt_def by simp

```

```

lemma not_eq_newpkt_rrep [simp]: " $\neg \text{eq\_newpkt } (\text{Rrep } \text{hops } \text{dip } \text{dsn } \text{oip } \text{sip})"$ 
  unfolding eq_newpkt_def by simp

```

```

lemma not_eq_newpkt_rerr [simp]: "¬eq_newpkt (Rerr dests sip)"
  unfolding eq_newpkt_def by simp

lemma not_eq_newpkt_pkt [simp]: "¬eq_newpkt (Pkt d dip sip)"
  unfolding eq_newpkt_def by simp

definition pkt :: "data × ip × ip ⇒ msg"
  where "pkt ≡ λ(d, dip, sip). Pkt d dip sip"

lemma pkt_simp [simp]:
  "pkt(d, dip, sip) = Pkt d dip sip"
  unfolding pkt_def by simp

end

```

## 5.3 The AODV protocol

```

theory E_Aodv
imports E_Aodv_Data E_Aodv_Message
  "../.../AWN/AWN_SOS_Labels" "../.../AWN/AWN_Invariants"
begin

```

### 5.3.1 Data state

```

record state =
  ip      :: "ip"
  sn      :: "sqn"
  rt      :: "rt"
  rreqs   :: "(ip × sqn) set"
  store   :: "store"

  msg     :: "msg"
  data    :: "data"
  dests   :: "ip → sqn"

  dip     :: "ip"
  oip     :: "ip"
  hops    :: "nat"
  dsn     :: "sqn"
  dsk     :: "k"
  osn     :: "sqn"
  sip     :: "ip"
  handled:: "bool"

abbreviation aodv_init :: "ip ⇒ state"
where "aodv_init i ≡ (|
  ip = i,
  sn = 1,
  rt = empty,
  rreqs = {},
  store = empty,

  msg = (SOME x. True),
  data = (SOME x. True),
  dests = (SOME x. True),

  dip = (SOME x. True),
  oip = (SOME x. True),
  hops = (SOME x. True),
  dsn = (SOME x. True),
  dsk = (SOME x. True),
  osn = (SOME x. True),
  sip = (SOME x. x ≠ i),
  handled = (SOME x. True)
)"

```



)"

lemma some\_neq\_not\_eq [simp]: " $\neg((\text{SOME } x :: \text{nat}. x \neq i) = i)$ "  
by (subst some\_eq\_ex) (metis zero\_neq\_numeral)

definition clear\_locals :: "state  $\Rightarrow$  state"

where "clear\_locals  $\xi = \xi$  (  
msg := (SOME x. True),  
data := (SOME x. True),  
dests := (SOME x. True),  
  
dip := (SOME x. True),  
oip := (SOME x. True),  
hops := (SOME x. True),  
dsn := (SOME x. True),  
dsk := (SOME x. True),  
osn := (SOME x. True),  
sip := (SOME x.  $x \neq \text{ip } \xi$ ),  
handled := (SOME x. True)  
)"

lemma clear\_locals\_sip\_not\_ip [simp]: " $\neg(\text{sip } (\text{clear\_locals } \xi) = \text{ip } \xi)$ "  
unfolding clear\_locals\_def by simp

lemma clear\_locals\_but\_not\_globals [simp]:

"ip (clear\_locals  $\xi$ ) = ip  $\xi$ "  
"sn (clear\_locals  $\xi$ ) = sn  $\xi$ "  
"rt (clear\_locals  $\xi$ ) = rt  $\xi$ "  
"rreqs (clear\_locals  $\xi$ ) = rreqs  $\xi$ "  
"store (clear\_locals  $\xi$ ) = store  $\xi$ "  
unfolding clear\_locals\_def by auto

### 5.3.2 Auxilliary message handling definitions

definition is\_newpkt

where "is\_newpkt  $\xi \equiv \text{case } \text{msg } \xi \text{ of}$   
Newpkt data' dip'  $\Rightarrow \{ \xi(\text{data} := \text{data}', \text{dip} := \text{dip}') \}$   
| \_  $\Rightarrow \{ \}$ "

definition is\_pkt

where "is\_pkt  $\xi \equiv \text{case } \text{msg } \xi \text{ of}$   
Pkt data' dip' oip'  $\Rightarrow \{ \xi(\text{data} := \text{data}', \text{dip} := \text{dip}', \text{oip} := \text{oip}') \}$   
| \_  $\Rightarrow \{ \}$ "

definition is\_rreq

where "is\_rreq  $\xi \equiv \text{case } \text{msg } \xi \text{ of}$   
Rreq hops' dip' dsn' dsk' oip' osn' sip' handled'  $\Rightarrow$   
 $\{ \xi(\text{hops} := \text{hops}', \text{dip} := \text{dip}', \text{dsn} := \text{dsn}',$   
 $\text{dsk} := \text{dsk}', \text{oip} := \text{oip}', \text{osn} := \text{osn}', \text{sip} := \text{sip}',$   
 $\text{handled} := \text{handled}') \}$   
| \_  $\Rightarrow \{ \}$ "

lemma is\_rreq\_asm [dest!]:

assumes " $\xi' \in \text{is\_rreq } \xi$ "  
shows " $(\exists \text{hops}' \text{dip}' \text{dsn}' \text{dsk}' \text{oip}' \text{osn}' \text{sip}' \text{handled}'.$   
 $\text{msg } \xi = \text{Rreq } \text{hops}' \text{dip}' \text{dsn}' \text{dsk}' \text{oip}' \text{osn}' \text{sip}' \text{handled}' \wedge$   
 $\xi' = \xi(\text{hops} := \text{hops}', \text{dip} := \text{dip}', \text{dsn} := \text{dsn}',$   
 $\text{dsk} := \text{dsk}', \text{oip} := \text{oip}', \text{osn} := \text{osn}', \text{sip} := \text{sip}',$   
 $\text{handled} := \text{handled}')$ "

using assms unfolding is\_rreq\_def

by (cases "msg  $\xi$ ") simp\_all

definition is\_rrep

where "is\_rrep  $\xi \equiv \text{case } \text{msg } \xi \text{ of}$

```

      Rrep hops' dip' dsn' oip' sip' ⇒
      { ξ(| hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' |) }
| _ ⇒ {}"

```

```

lemma is_rrep_asm [dest!]:
  assumes "ξ' ∈ is_rrep ξ"
  shows "(∃ hops' dip' dsn' oip' sip'.
    msg ξ = Rrep hops' dip' dsn' oip' sip' ∧
    ξ' = ξ(| hops := hops', dip := dip', dsn := dsn', oip := oip', sip := sip' |))"
  using assms unfolding is_rrep_def
  by (cases "msg ξ") simp_all

```

```

definition is_rerr
where "is_rerr ξ ≡ case msg ξ of
      Rerr dests' sip' ⇒ { ξ(| dests := dests', sip := sip' |) }
| _ ⇒ {}"

```

```

lemma is_rerr_asm [dest!]:
  assumes "ξ' ∈ is_rerr ξ"
  shows "(∃ dests' sip'.
    msg ξ = Rerr dests' sip' ∧
    ξ' = ξ(| dests := dests', sip := sip' |))"
  using assms unfolding is_rerr_def
  by (cases "msg ξ") simp_all

```

```

lemmas is_msg_defs =
  is_rerr_def is_rrep_def is_rreq_def is_pkt_def is_newpkt_def

```

```

lemma is_msg_inv_ip [simp]:
  "ξ' ∈ is_rerr ξ ⇒ ip ξ' = ip ξ"
  "ξ' ∈ is_rrep ξ ⇒ ip ξ' = ip ξ"
  "ξ' ∈ is_rreq ξ ⇒ ip ξ' = ip ξ"
  "ξ' ∈ is_pkt ξ ⇒ ip ξ' = ip ξ"
  "ξ' ∈ is_newpkt ξ ⇒ ip ξ' = ip ξ"
  unfolding is_msg_defs
  by (cases "msg ξ", clarsimp+)+

```

```

lemma is_msg_inv_sn [simp]:
  "ξ' ∈ is_rerr ξ ⇒ sn ξ' = sn ξ"
  "ξ' ∈ is_rrep ξ ⇒ sn ξ' = sn ξ"
  "ξ' ∈ is_rreq ξ ⇒ sn ξ' = sn ξ"
  "ξ' ∈ is_pkt ξ ⇒ sn ξ' = sn ξ"
  "ξ' ∈ is_newpkt ξ ⇒ sn ξ' = sn ξ"
  unfolding is_msg_defs
  by (cases "msg ξ", clarsimp+)+

```

```

lemma is_msg_inv_rt [simp]:
  "ξ' ∈ is_rerr ξ ⇒ rt ξ' = rt ξ"
  "ξ' ∈ is_rrep ξ ⇒ rt ξ' = rt ξ"
  "ξ' ∈ is_rreq ξ ⇒ rt ξ' = rt ξ"
  "ξ' ∈ is_pkt ξ ⇒ rt ξ' = rt ξ"
  "ξ' ∈ is_newpkt ξ ⇒ rt ξ' = rt ξ"
  unfolding is_msg_defs
  by (cases "msg ξ", clarsimp+)+

```

```

lemma is_msg_inv_rreqs [simp]:
  "ξ' ∈ is_rerr ξ ⇒ rreqs ξ' = rreqs ξ"
  "ξ' ∈ is_rrep ξ ⇒ rreqs ξ' = rreqs ξ"
  "ξ' ∈ is_rreq ξ ⇒ rreqs ξ' = rreqs ξ"
  "ξ' ∈ is_pkt ξ ⇒ rreqs ξ' = rreqs ξ"
  "ξ' ∈ is_newpkt ξ ⇒ rreqs ξ' = rreqs ξ"
  unfolding is_msg_defs
  by (cases "msg ξ", clarsimp+)+

```

```

lemma is_msg_inv_store [simp]:
  "ξ' ∈ is_rerr ξ  ⇒ store ξ' = store ξ"
  "ξ' ∈ is_rrep ξ  ⇒ store ξ' = store ξ"
  "ξ' ∈ is_rreq ξ  ⇒ store ξ' = store ξ"
  "ξ' ∈ is_pkt ξ   ⇒ store ξ' = store ξ"
  "ξ' ∈ is_newpkt ξ ⇒ store ξ' = store ξ"
  unfolding is_msg_defs
  by (cases "msg ξ", clarsimp+)+

```

```

lemma is_msg_inv_sip [simp]:
  "ξ' ∈ is_pkt ξ   ⇒ sip ξ' = sip ξ"
  "ξ' ∈ is_newpkt ξ ⇒ sip ξ' = sip ξ"
  unfolding is_msg_defs
  by (cases "msg ξ", clarsimp+)+

```

### 5.3.3 The protocol process

```

datatype pseqp =
  PAadv
  | PNewPkt
  | PPkt
  | PRreq
  | PRrep
  | PRerr

```

```

fun nat_of_seqp :: "pseqp ⇒ nat"

```

```

where
  "nat_of_seqp PAadv = 1"
  | "nat_of_seqp PPkt = 2"
  | "nat_of_seqp PNewPkt = 3"
  | "nat_of_seqp PRreq = 4"
  | "nat_of_seqp PRrep = 5"
  | "nat_of_seqp PRerr = 6"

```

```

instantiation "pseqp" :: ord

```

```

begin

```

```

definition less_eq_seqp [iff]: "l1 ≤ l2 = (nat_of_seqp l1 ≤ nat_of_seqp l2)"

```

```

definition less_seqp [iff]: "l1 < l2 = (nat_of_seqp l1 < nat_of_seqp l2)"

```

```

instance ..

```

```

end

```

```

abbreviation AODV

```

```

where

```

```

  "AODV ≡ λ_. [[clear_locals]] call(PAadv)"

```

```

abbreviation PKT

```

```

where

```

```

  "PKT args ≡

```

```

    [[ξ. let (data, dip, oip) = args ξ in
      (clear_locals ξ) (| data := data, dip := dip, oip := oip |)]
    call(PPkt)"

```

```

abbreviation NEWPKT

```

```

where

```

```

  "NEWPKT args ≡
    [[ξ. let (data, dip) = args ξ in
      (clear_locals ξ) (| data := data, dip := dip |)]
    call(PNewPkt)"

```

```

abbreviation RREQ

```

```

where

```

```

  "RREQ args ≡
    [[ξ. let (hops, dip, dsn, dsk, oip, osn, sip, handled) = args ξ in
      (clear_locals ξ) (| hops := hops, dip := dip,

```

```

        dsn := dsn, dsk := dsk, oip := oip,
        osn := osn, sip := sip, handled := handled ])

call(PRreq)"

abbreviation RREP
where
  "RREP args ≡
  [[ξ. let (hops, dip, dsn, oip, sip) = args ξ in
    (clear_locals ξ) (| hops := hops, dip := dip, dsn := dsn,
      oip := oip, sip := sip |)]

  call(PRrep)"

abbreviation RERR
where
  "RERR args ≡
  [[ξ. let (dests, sip) = args ξ in
    (clear_locals ξ) (| dests := dests, sip := sip |)]

  call(PRerr)"

fun ΓAODV :: "(state, msg, pseqp, pseqp label) seqp_env"
where
  "ΓAODV PAodv = labelled PAodv (
    receive(λmsg' ξ. ξ (| msg := msg' |)).
    (
      ⟨is_newpkt⟩ NEWPKT(λξ. (data ξ, ip ξ))
      ⊕ ⟨is_pkt⟩ PKT(λξ. (data ξ, dip ξ, oip ξ))
      ⊕ ⟨is_rreq⟩
        [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ) |)]
        RREQ(λξ. (hops ξ, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, sip ξ, handled ξ))
      ⊕ ⟨is_rrep⟩
        [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ) |)]
        RREP(λξ. (hops ξ, dip ξ, dsn ξ, oip ξ, sip ξ))
      ⊕ ⟨is_rerr⟩
        [[ξ. ξ (|rt := update (rt ξ) (sip ξ) (0, unk, val, 1, sip ξ) |)]
        RERR(λξ. (dests ξ, sip ξ))
    )
    ⊕ ⟨λξ. { ξ (| dip := dip |) | dip. dip ∈ qD(store ξ) ∩ vD(rt ξ) }⟩
      [[ξ. ξ (| data := hd(σqueue(store ξ, dip ξ)) |)]
      unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, ip ξ)).
      [[ξ. ξ (| store := the (drop (dip ξ) (store ξ)) |)]
      AODV()
      ▷ [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
        then Some (inc (sqn (rt ξ) rip)) else None) |)]
        [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]
        [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]
        broadcast(λξ. rerr(dests ξ, ip ξ)). AODV()
    ⊕ ⟨λξ. { ξ (| dip := dip |)
      | dip. dip ∈ qD(store ξ) - vD(rt ξ) ∧ the (σp-flag(store ξ, dip)) = req }⟩
      [[ξ. ξ (| store := unsetRRF (store ξ) (dip ξ) |)]
      [[ξ. ξ (| sn := inc (sn ξ) |)]
      [[ξ. ξ (| rreqs := rreqs ξ ∪ {(ip ξ, sn ξ)} |)]
      broadcast(λξ. rreq(0, dip ξ, sqn (rt ξ) (dip ξ), sqnf (rt ξ) (dip ξ), ip ξ, sn ξ,
        ip ξ, False)). AODV()]"

| "ΓAODV PNewPkt = labelled PNewPkt (
  ⟨ξ. dip ξ = ip ξ⟩
  deliver(λξ. data ξ).AODV()
  ⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
  [[ξ. ξ (| store := add (data ξ) (dip ξ) (store ξ) |)]
  AODV()]"

| "ΓAODV PPkt = labelled PPkt (
  ⟨ξ. dip ξ = ip ξ⟩
  deliver(λξ. data ξ).AODV()
  ⊕ ⟨ξ. dip ξ ≠ ip ξ⟩

```

```

(
  ⟨ξ. dip ξ ∈ vD (rt ξ)⟩
  unicast(λξ. the (nhop (rt ξ) (dip ξ)), λξ. pkt(data ξ, dip ξ, oip ξ)).AODV()
  ▷
    [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (dip ξ))
      then Some (inc (sqn (rt ξ) rip)) else None) |]]
    [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |]]
    [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |]]
    broadcast(λξ. rerr(dests ξ, ip ξ)).AODV()
  ⊕ ⟨ξ. dip ξ ∉ vD (rt ξ)⟩
  (
    ⟨ξ. dip ξ ∈ iD (rt ξ)⟩
    broadcast(λξ. rerr([dip ξ ↦ sqn (rt ξ) (dip ξ)], ip ξ)). AODV()
  ⊕ ⟨ξ. dip ξ ∉ iD (rt ξ)⟩
    AODV()
  )
  )
))"

| "ΓAODV PRreq = labelled PRreq (
  ⟨ξ. (oip ξ, osn ξ) ∈ rreqs ξ⟩
  AODV()
  ⊕ ⟨ξ. (oip ξ, osn ξ) ∉ rreqs ξ⟩
  [[ξ. ξ (| rt := update (rt ξ) (oip ξ) (osn ξ, kno, val, hops ξ + 1, sip ξ) |]]
  [[ξ. ξ (| rreqs := rreqs ξ ∪ {(oip ξ, osn ξ)} |]]
  (
    ⟨ξ. handled ξ = False⟩
    (
      ⟨ξ. dip ξ = ip ξ⟩
      [[ξ. ξ (| sn := max (sn ξ) (dsn ξ) |]]
      unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(0, dip ξ, sn ξ, oip ξ, ip ξ)).
      broadcast(λξ. rreq(hops ξ + 1, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, ip ξ, True)).
      AODV()
    ▷
      [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
        then Some (inc (sqn (rt ξ) rip)) else None) |]]
      [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |]]
      [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |]]
      broadcast(λξ. rerr(dests ξ, ip ξ)).AODV()
    ⊕ ⟨ξ. dip ξ ≠ ip ξ⟩
    (
      ⟨ξ. dip ξ ∈ vD (rt ξ) ∧ dsn ξ ≤ sqn (rt ξ) (dip ξ) ∧ sqnf (rt ξ) (dip ξ) = kno⟩
      unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(the (dhops (rt ξ) (dip ξ)), dip ξ,
        sqn (rt ξ) (dip ξ), oip ξ, ip ξ)).
      broadcast(λξ. rreq(hops ξ + 1, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, ip ξ, True)).
      AODV()
    ▷
      [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
        then Some (inc (sqn (rt ξ) rip)) else None) |]]
      [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |]]
      [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |]]
      broadcast(λξ. rerr(dests ξ, ip ξ)).AODV()
    ⊕ ⟨ξ. dip ξ ∉ vD (rt ξ) ∨ sqn (rt ξ) (dip ξ) < dsn ξ ∨ sqnf (rt ξ) (dip ξ) = unk⟩
      broadcast(λξ. rreq(hops ξ + 1, dip ξ, max (sqn (rt ξ) (dip ξ)) (dsn ξ),
        dsk ξ, oip ξ, osn ξ, ip ξ, False)).
      AODV()
    )
  )
  )
  ⊕ ⟨ξ. handled ξ = True⟩
  broadcast(λξ. rreq(hops ξ + 1, dip ξ, dsn ξ, dsk ξ, oip ξ, osn ξ, ip ξ, True)).
  AODV()
  )
))"

| "ΓAODV PRrep = labelled PRrep (
  [[ξ. ξ (| rt := update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ) | ] ]

```

```

(
  ⟨ξ. oip ξ = ip ξ ⟩
  AODV()
⊕ ⟨ξ. oip ξ ≠ ip ξ ⟩
(
  ⟨ξ. oip ξ ∈ vD (rt ξ) ∧ dip ξ ∈ vD (rt ξ)⟩
  unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(the (dhops (rt ξ) (dip ξ)), dip ξ,
    sqn (rt ξ) (dip ξ), oip ξ, ip ξ)).
  AODV()
▷
  [[ξ. ξ (| dests := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
    then Some (inc (sqn (rt ξ) rip)) else None) |])]
  [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]]
  [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]]
  broadcast(λξ. rerr(dests ξ, ip ξ)).AODV()
⊕ ⟨ξ. oip ξ ∉ vD (rt ξ) ∨ dip ξ ∉ vD (rt ξ)⟩
  AODV()
)
)
)"

/ "ΓAODV PRerr = labelled PRerr (
  [[ξ. ξ (| dests := (λrip. case (dests ξ) rip of None ⇒ None
    | Some rsn ⇒ if rip ∈ vD (rt ξ) ∧ the (nhop (rt ξ) rip) = sip ξ
      ∧ sqn (rt ξ) rip < rsn then Some rsn else None) |])]
  [[ξ. ξ (| rt := invalidate (rt ξ) (dests ξ) |)]]
  [[ξ. ξ (| store := setRRF (store ξ) (dests ξ) |)]]
  (
    ⟨ξ. dests ξ ≠ Map.empty⟩
    broadcast(λξ. rerr(dests ξ, ip ξ)). AODV()
  ⊕ ⟨ξ. dests ξ = Map.empty⟩
    AODV()
  )
)"

```

```

declare ΓAODV.simps [simp del, code del]
lemmas ΓAODV.simps [simp, code] = ΓAODV.simps [simplified]

```

```

fun ΓAODV.skeleton
where
  "ΓAODV.skeleton PAodv = seqp_skeleton (ΓAODV PAodv)"
  | "ΓAODV.skeleton PNewPkt = seqp_skeleton (ΓAODV PNewPkt)"
  | "ΓAODV.skeleton PPkt = seqp_skeleton (ΓAODV PPkt)"
  | "ΓAODV.skeleton PRreq = seqp_skeleton (ΓAODV PRreq)"
  | "ΓAODV.skeleton PRrep = seqp_skeleton (ΓAODV PRrep)"
  | "ΓAODV.skeleton PRerr = seqp_skeleton (ΓAODV PRerr)"

```

```

lemma ΓAODV.skeleton_wf [simp]:
  "wellformed ΓAODV.skeleton"
proof (rule, intro allI)
  fix pn pn'
  show "call(pn') ∉ stermsl (ΓAODV.skeleton pn)"
  by (cases pn) simp_all
qed

```

```

declare ΓAODV.skeleton.simps [simp del, code del]
lemmas ΓAODV.skeleton.simps [simp, code]
  = ΓAODV.skeleton.simps [simplified ΓAODV.simps seqp_skeleton.simps]

```

```

lemma aodv_proc_cases [dest]:
  fixes p pn
  shows "p ∈ ctermsl (ΓAODV pn) ⇒
    (p ∈ ctermsl (ΓAODV PAodv) ∨
     p ∈ ctermsl (ΓAODV PNewPkt) ∨

```

```

      p ∈ cterms1 (ΓAODV PPkt) ∨
      p ∈ cterms1 (ΓAODV PRreq) ∨
      p ∈ cterms1 (ΓAODV PRrep) ∨
      p ∈ cterms1 (ΓAODV PRerr))"

using assms
by (cases pn) simp_all

definition σAODV :: "ip ⇒ (state × (state, msg, pseq, pseq label) seq) set"
where "σAODV i ≡ {(aodv_init i, ΓAODV PAodv)}"

abbreviation paodv
  :: "ip ⇒ (state × (state, msg, pseq, pseq label) seq, msg seq_action) automaton"
where
  "paodv i ≡ (| init = σAODV i, trans = seqp_sos ΓAODV |)"

lemma aodv_trans: "trans (paodv i) = seqp_sos ΓAODV"
  by simp

lemma aodv_control_within [simp]: "control_within ΓAODV (init (paodv i))"
  unfolding σAODV_def by (rule control_withinI) (auto simp del: ΓAODV_simps)

lemma aodv_wf [simp]:
  "wellformed ΓAODV"
proof (rule, intro allI)
  fix pn pn'
  show "call(pn') ∉ sterms1 (ΓAODV pn)"
    by (cases pn) simp_all
qed

lemmas aodv_labels_not_empty [simp] = labels_not_empty [OF aodv_wf]

lemma aodv_ex_label [intro]: "∃ l. l ∈ labels ΓAODV p"
  by (metis aodv_labels_not_empty all_not_in_conv)

lemma aodv_ex_labelE [elim]:
  assumes "∀ l ∈ labels ΓAODV p. P l p"
  and "∃ p l. P l p ⇒ Q"
  shows "Q"
  using assms by (metis aodv_ex_label)

lemma aodv_simple_labels [simp]: "simple_labels ΓAODV"
proof
  fix pn p
  assume "p ∈ subterms(ΓAODV pn)"
  thus "∃ l. l ∈ labels ΓAODV p = {l}"
    by (cases pn) (simp_all cong: seqp_congs | elim disjE)+
qed

lemma σAODV_labels [simp]: "(ξ, p) ∈ σAODV i ⇒ labels ΓAODV p = {PAodv-:0}"
  unfolding σAODV_def by simp

lemma aodv_init_kD_empty [simp]:
  "(ξ, p) ∈ σAODV i ⇒ kD (rt ξ) = {}"
  unfolding σAODV_def kD_def by simp

lemma aodv_init_sip_not_ip [simp]: "¬(sip (aodv_init i) = i)" by simp

lemma aodv_init_sip_not_ip' [simp]:
  assumes "(ξ, p) ∈ σAODV i"
  shows "sip ξ ≠ ip ξ"
  using assms unfolding σAODV_def by simp

lemma aodv_init_sip_not_i [simp]:
  assumes "(ξ, p) ∈ σAODV i"

```

```

shows "sip  $\xi \neq i$ "
using assms unfolding  $\sigma_{AODV\_def}$  by simp

```

```

lemma clear_locals_sip_not_ip':
  assumes "ip  $\xi = i$ "
  shows " $\neg$ (sip (clear_locals  $\xi$ ) = i)"
  using assms by auto

```

Stop the simplifier from descending into process terms.

```

declare seq_congs [cong]

```

Configure the main invariant tactic for AODV.

```

declare
   $\Gamma_{AODV\_simps}$  [cterms_env]
  aadv_proc_cases [ctermstl_cases]
  seq_invariant_ctermstl [OF aadv_wf aadv_control_within aadv_simple_labels aadv_trans,
                           cterms_intros]
  seq_step_invariant_ctermstl [OF aadv_wf aadv_control_within aadv_simple_labels aadv_trans,
                                cterms_intros]

end

```

## 5.4 Invariant assumptions and properties

```

theory E_Aadv_Predicates
imports E_Aadv
begin

```

Definitions for expression assumptions on incoming messages and properties of outgoing messages.

```

abbreviation not_Pkt :: "msg  $\Rightarrow$  bool"
where "not_Pkt m  $\equiv$  case m of Pkt _ _ _  $\Rightarrow$  False | _  $\Rightarrow$  True"

```

```

definition msg_sender :: "msg  $\Rightarrow$  ip"
where "msg_sender m  $\equiv$  case m of Rreq _ _ _ _ _ ipc _  $\Rightarrow$  ipc
      | Rrep _ _ _ _ ipc  $\Rightarrow$  ipc
      | Rerr _ ipc  $\Rightarrow$  ipc
      | Pkt _ _ ipc  $\Rightarrow$  ipc"

```

```

lemma msg_sender_simps [simp]:
  " $\bigwedge$ hops dip dsn dsk oip osn sip handled.
    msg_sender (Rreq hops dip dsn dsk oip osn sip handled) = sip"
  " $\bigwedge$ hops dip dsn oip sip. msg_sender (Rrep hops dip dsn oip sip) = sip"
  " $\bigwedge$ dests sip. msg_sender (Rerr dests sip) = sip"
  " $\bigwedge$ dip sip. msg_sender (Pkt d dip sip) = sip"
  unfolding msg_sender_def by simp_all

```

```

definition msg_zhops :: "msg  $\Rightarrow$  bool"
where "msg_zhops m  $\equiv$  case m of
      Rreq hopsdip dpc _ _ oipc _ sipc _  $\Rightarrow$  hopsdip = 0  $\longrightarrow$  oipc = sipc
      | Rrep hopsdip dpc _ _ sipc  $\Rightarrow$  hopsdip = 0  $\longrightarrow$  dpc = sipc
      | _  $\Rightarrow$  True"

```

```

lemma msg_zhops_simps [simp]:
  " $\bigwedge$ hops dip dsn dsk oip osn sip handled.
    msg_zhops (Rreq hops dip dsn dsk oip osn sip handled) = (hops = 0  $\longrightarrow$  oip = sip)"
  " $\bigwedge$ hops dip dsn oip sip. msg_zhops (Rrep hops dip dsn oip sip) = (hops = 0  $\longrightarrow$  dip = sip)"
  " $\bigwedge$ dests sip. msg_zhops (Rerr dests sip) = True"
  " $\bigwedge$ dip. msg_zhops (Newpkt d dip) = True"
  " $\bigwedge$ dip sip. msg_zhops (Pkt d dip sip) = True"
  unfolding msg_zhops_def by simp_all

```

```

definition rreq_rrep_sn :: "msg  $\Rightarrow$  bool"
where "rreq_rrep_sn m  $\equiv$  case m of Rreq _ _ _ _ _ osnc _ _  $\Rightarrow$  osnc  $\geq$  1"

```



```

| Rrep _ _ dsnc _ _  $\Rightarrow$  dsnc  $\geq$  1
| _  $\Rightarrow$  True"

```

lemma rreq\_rrep\_sn\_simps [simp]:

```

" $\wedge$ hops dip dsn dsk oip osn sip handled.
  rreq_rrep_sn (Rreq hops dip dsn dsk oip osn sip handled) = (osn  $\geq$  1)"
" $\wedge$ hops dip dsn oip sip. rreq_rrep_sn (Rrep hops dip dsn oip sip) = (dsn  $\geq$  1)"
" $\wedge$ dests sip. rreq_rrep_sn (Rerr dests sip) = True"
" $\wedge$ d dip. rreq_rrep_sn (Newpkt d dip) = True"
" $\wedge$ d dip sip. rreq_rrep_sn (Pkt d dip sip) = True"
unfolding rreq_rrep_sn_def by simp_all

```

definition rreq\_rrep\_fresh :: "rt  $\Rightarrow$  msg  $\Rightarrow$  bool"

```

where "rreq_rrep_fresh crt m  $\equiv$  case m of Rreq hopsc _ _ _ oipc osnc ipcc _  $\Rightarrow$  (ipcc  $\neq$  oipc  $\longrightarrow$ 
  oipc $\in$ kD(crt)  $\wedge$  (sqn crt oipc > osnc
     $\vee$  (sqn crt oipc = osnc
       $\wedge$  the (dhops crt oipc)  $\leq$  hopsc
       $\wedge$  the (flag crt oipc) = val)))
| Rrep hopsc dipc dsnc _ ipcc  $\Rightarrow$  (ipcc  $\neq$  dipc  $\longrightarrow$ 
  dipc $\in$ kD(crt)
   $\wedge$  sqn crt dipc = dsnc
   $\wedge$  the (dhops crt dipc) = hopsc
   $\wedge$  the (flag crt dipc) = val)
| _  $\Rightarrow$  True"

```

lemma rreq\_rrep\_fresh [simp]:

```

" $\wedge$ hops dip dsn dsk oip osn sip handled.
  rreq_rrep_fresh crt (Rreq hops dip dsn dsk oip osn sip handled) =
  (sip  $\neq$  oip  $\longrightarrow$  oip $\in$ kD(crt)
     $\wedge$  (sqn crt oip > osn
       $\vee$  (sqn crt oip = osn
         $\wedge$  the (dhops crt oip)  $\leq$  hops
         $\wedge$  the (flag crt oip) = val)))"
" $\wedge$ hops dip dsn oip sip. rreq_rrep_fresh crt (Rrep hops dip dsn oip sip) =
  (sip  $\neq$  dip  $\longrightarrow$  dip $\in$ kD(crt)
     $\wedge$  sqn crt dip = dsn
     $\wedge$  the (dhops crt dip) = hops
     $\wedge$  the (flag crt dip) = val)"
" $\wedge$ dests sip. rreq_rrep_fresh crt (Rerr dests sip) = True"
" $\wedge$ d dip. rreq_rrep_fresh crt (Newpkt d dip) = True"
" $\wedge$ d dip sip. rreq_rrep_fresh crt (Pkt d dip sip) = True"
unfolding rreq_rrep_fresh_def by simp_all

```

definition rerr\_invalid :: "rt  $\Rightarrow$  msg  $\Rightarrow$  bool"

```

where "rerr_invalid crt m  $\equiv$  case m of Rerr destsc _  $\Rightarrow$  ( $\forall$ ripc $\in$ dom(destsc).
  (ripc $\in$ iD(crt)  $\wedge$  the (destsc ripc) = sqn crt ripc))
| _  $\Rightarrow$  True"

```

lemma rerr\_invalid [simp]:

```

" $\wedge$ hops dip dsn dsk oip osn sip handled.
  rerr_invalid crt (Rreq hops dip dsn dsk oip osn sip handled) = True"
" $\wedge$ hops dip dsn oip sip. rerr_invalid crt (Rrep hops dip dsn oip sip) = True"
" $\wedge$ dests sip. rerr_invalid crt (Rerr dests sip) = ( $\forall$ ripc $\in$ dom(dests).
  ripc $\in$ iD(crt)  $\wedge$  the (dests ripc) = sqn crt ripc)"
" $\wedge$ d dip. rerr_invalid crt (Newpkt d dip) = True"
" $\wedge$ d dip sip. rerr_invalid crt (Pkt d dip sip) = True"
unfolding rerr_invalid_def by simp_all

```

definition

```

initmissing :: "(nat  $\Rightarrow$  state option)  $\times$  'a  $\Rightarrow$  (nat  $\Rightarrow$  state)  $\times$  'a"

```

where

```

"initmissing  $\sigma$  = ( $\lambda$ i. case (fst  $\sigma$ ) i of None  $\Rightarrow$  aadv_init i | Some s  $\Rightarrow$  s, snd  $\sigma$ )"

```

lemma not\_in\_net\_ips\_fst\_init\_missing [simp]:

```

assumes "i  $\notin$  net_ips  $\sigma$ "
shows "fst (initmissing (netgmap fst  $\sigma$ )) i = aadv_init i"
using assms unfolding initmissing_def by simp

```

```

lemma fst_initmissing_netgmap_pair_fst [simp]:
  "fst (initmissing (netgmap ( $\lambda$ (p, q). (fst (id p), snd (id p), q)) s))
    = fst (initmissing (netgmap fst s))"
  unfolding initmissing_def by auto

```

We introduce a streamlined alternative to `initmissing` with `netgmap` to simplify invariant statements and thus facilitate their comprehension and presentation.

```

lemma fst_initmissing_netgmap_default_aadv_init_netlift:
  "fst (initmissing (netgmap fst s)) = default aadv_init (netlift fst s)"
  unfolding initmissing_def default_def
  by (simp add: fst_netgmap_netlift del: One_nat_def)

```

```

definition
  netglobal :: "(nat  $\Rightarrow$  state)  $\Rightarrow$  bool  $\Rightarrow$  ((state  $\times$  'b)  $\times$  'c) net_state  $\Rightarrow$  bool"
where
  "netglobal P  $\equiv$  ( $\lambda$ s. P (default aadv_init (netlift fst s)))"

```

end

## 5.5 Quality relations between routes

```

theory E_Fresher
imports E_Aadv_Data
begin

```

### 5.5.1 Net sequence numbers

On individual routes

```

definition
  nsqn_r :: "r  $\Rightarrow$  sqn"
where
  "nsqn_r r  $\equiv$  if  $\pi_4$ (r) = val  $\vee$   $\pi_2$ (r) = 0 then  $\pi_2$ (r) else ( $\pi_2$ (r) - 1)"

```

```

lemma nsqn_r_def':
  "nsqn_r r = (if  $\pi_4$ (r) = inv then  $\pi_2$ (r) - 1 else  $\pi_2$ (r))"
  unfolding nsqn_r_def by simp

```

```

lemma nsqn_r_zero [simp]:
  " $\bigwedge$ dsn dsk flag hops nhip. nsqn_r (0, dsk, flag, hops, nhip) = 0"
  unfolding nsqn_r_def by clarsimp

```

```

lemma nsqn_r_val [simp]:
  " $\bigwedge$ dsn dsk hops nhip. nsqn_r (dsn, dsk, val, hops, nhip) = dsn"
  unfolding nsqn_r_def by clarsimp

```

```

lemma nsqn_r_inv [simp]:
  " $\bigwedge$ dsn dsk hops nhip. nsqn_r (dsn, dsk, inv, hops, nhip) = dsn - 1"
  unfolding nsqn_r_def by clarsimp

```

```

lemma nsqn_r_lte_dsn [simp]:
  " $\bigwedge$ dsn dsk flag hops nhip. nsqn_r (dsn, dsk, flag, hops, nhip)  $\leq$  dsn"
  unfolding nsqn_r_def by clarsimp

```

On routes in routing tables

```

definition
  nsqn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  sqn"
where
  "nsqn  $\equiv$   $\lambda$ rt dip. case  $\sigma_{route}$ (rt, dip) of None  $\Rightarrow$  0 | Some r  $\Rightarrow$  nsqn_r(r)"

```

```

lemma nsqn_sqn_def:
  " $\bigwedge$ rt dip. nsqn rt dip = (if flag rt dip = Some val  $\vee$  sqn rt dip = 0
    then sqn rt dip else sqn rt dip - 1)"
  unfolding nsqn_def sqn_def by (clarsimp split: option.split) auto

lemma not_in_kD_nsqn [simp]:
  assumes "dip  $\notin$  kD(rt)"
  shows "nsqn rt dip = 0"
  using assms unfolding nsqn_def by simp

lemma kD_nsqn:
  assumes "dip  $\in$  kD(rt)"
  shows "nsqn rt dip = nsqnr(the ( $\sigma_{route}$ (rt, dip)))"
  using assms [THEN kD_Some] unfolding nsqn_def by clarsimp

lemma nsqn_r_flag_pred [simp, intro]:
  fixes dsn dsk flag hops nhip pre
  assumes "P (nsqnr (dsn, dsk, val, hops, nhip))"
    and "P (nsqnr (dsn, dsk, inv, hops, nhip))"
  shows "P (nsqnr (dsn, dsk, flag, hops, nhip))"
  using assms by (cases flag) auto

lemma sqn_nsqn:
  " $\bigwedge$ rt dip. sqn rt dip - 1  $\leq$  nsqn rt dip"
  unfolding sqn_def nsqn_def by (clarsimp split: option.split)

lemma nsqn_sqn: "nsqn rt dip  $\leq$  sqn rt dip"
  unfolding sqn_def nsqn_def by (cases "rt dip") auto

lemma val_nsqn_sqn [elim, simp]:
  assumes "ip  $\in$  kD(rt)"
    and "the (flag rt ip) = val"
  shows "nsqn rt ip = sqn rt ip"
  using assms unfolding nsqn_sqn_def by auto

lemma vD_nsqn_sqn [elim, simp]:
  assumes "ip  $\in$  vD(rt)"
  shows "nsqn rt ip = sqn rt ip"
  proof -
    from (ip  $\in$  vD(rt)) have "ip  $\in$  kD(rt)"
      and "the (flag rt ip) = val" by auto
    thus ?thesis ..
  qed

lemma inv_nsqn_sqn [elim, simp]:
  assumes "ip  $\in$  kD(rt)"
    and "the (flag rt ip) = inv"
  shows "nsqn rt ip = sqn rt ip - 1"
  using assms unfolding nsqn_sqn_def by auto

lemma iD_nsqn_sqn [elim, simp]:
  assumes "ip  $\in$  iD(rt)"
  shows "nsqn rt ip = sqn rt ip - 1"
  proof -
    from (ip  $\in$  iD(rt)) have "ip  $\in$  kD(rt)"
      and "the (flag rt ip) = inv" by auto
    thus ?thesis ..
  qed

lemma nsqn_update_changed_kno_val [simp]: " $\bigwedge$ rt ip dsn dsk hops nhip.
  rt  $\neq$  update rt ip (dsn, kno, val, hops, nhip)
 $\implies$  nsqn (update rt ip (dsn, kno, val, hops, nhip)) ip = dsn"
  using assms unfolding nsqn_r_def update_def

```

```

by (clarsimp simp: kD_nsqn split: option.split_asm option.split split_if_asm)
  (metis fun_upd_triv)

```

```

lemma nsqn_update_other [simp]:
  fixes dsn dsk flag hops dip nhip pre rt ip
  assumes "dip  $\neq$  ip"
  shows "nsqn (update rt ip (dsn, dsk, flag, hops, nhip)) dip = nsqn rt dip"
  using assms unfolding nsqn_def
  by (clarsimp split: option.split)

```

```

lemma nsqn_invalidate_eq:
  assumes "dip  $\in$  kD(rt)"
    and "dests dip = Some rsn"
  shows "nsqn (invalidate rt dests) dip = rsn - 1"
  using assms
  proof -
    from assms obtain dsk hops nhip
      where "invalidate rt dests dip = Some (rsn, dsk, inv, hops, nhip)"
    unfolding invalidate_def
    by auto
    moreover from  $\langle$ dip  $\in$  kD(rt) $\rangle$  have "dip  $\in$  kD(invalidate rt dests)" by simp
    ultimately show ?thesis
      using  $\langle$ dests dip = Some rsn $\rangle$  by simp
  qed

```

```

lemma nsqn_invalidate_other [simp]:
  assumes "dip  $\in$  kD(rt)"
    and "dip  $\notin$  dom dests"
  shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
  using assms by (clarsimp simp add: kD_nsqn)

```

## 5.5.2 Comparing routes

definition

```

fresher :: "r  $\Rightarrow$  r  $\Rightarrow$  bool" ("(_/  $\sqsubseteq$  _)" [51, 51] 50)

```

where

```

"fresher r r'  $\equiv$  ((nsqnr r < nsqnr r')  $\vee$  (nsqnr r = nsqnr r'  $\wedge$   $\pi_5$ (r)  $\geq$   $\pi_5$ (r')))"

```

```

lemma fresherI1 [intro]:
  assumes "nsqnr r < nsqnr r'"
  shows "r  $\sqsubseteq$  r'"
  unfolding fresher_def using assms by simp

```

```

lemma fresherI2 [intro]:
  assumes "nsqnr r = nsqnr r'"
    and " $\pi_5$ (r)  $\geq$   $\pi_5$ (r'"
  shows "r  $\sqsubseteq$  r'"
  unfolding fresher_def using assms by simp

```

```

lemma fresherI [intro]:
  assumes "(nsqnr r < nsqnr r')  $\vee$  (nsqnr r = nsqnr r'  $\wedge$   $\pi_5$ (r)  $\geq$   $\pi_5$ (r'))"
  shows "r  $\sqsubseteq$  r'"
  unfolding fresher_def using assms .

```

```

lemma fresherE [elim]:
  assumes "r  $\sqsubseteq$  r'"
    and "nsqnr r < nsqnr r'  $\implies$  P r r'"
    and "nsqnr r = nsqnr r'  $\wedge$   $\pi_5$ (r)  $\geq$   $\pi_5$ (r')  $\implies$  P r r'"
  shows "P r r'"
  using assms unfolding fresher_def by auto

```

```

lemma fresher_refl [simp]: "r  $\sqsubseteq$  r"
  unfolding fresher_def by simp

```

lemma fresher\_trans [elim, trans]:

"[[ x  $\sqsubseteq$  y; y  $\sqsubseteq$  z ]  $\implies$  x  $\sqsubseteq$  z"  
 unfolding fresher\_def by auto

lemma not\_fresher\_trans [elim, trans]:

"[[  $\neg(x \sqsubseteq y)$ ;  $\neg(z \sqsubseteq x)$  ]  $\implies$   $\neg(z \sqsubseteq y)$ "  
 unfolding fresher\_def by auto

lemma fresher\_dsn\_flag\_hops\_const [simp]:

fixes dsn dsk dsk' flag hops nhip nhip'  
 shows "(dsn, dsk, flag, hops, nhip)  $\sqsubseteq$  (dsn, dsk', flag, hops, nhip)"  
 unfolding fresher\_def by (cases flag) simp\_all

### 5.5.3 Comparing routing tables

definition

rt\_fresher :: "ip  $\Rightarrow$  rt  $\Rightarrow$  rt  $\Rightarrow$  bool"

where

"rt\_fresher  $\equiv$   $\lambda$ dip rt rt'. (the ( $\sigma_{route}(rt, dip)$ ))  $\sqsubseteq$  (the ( $\sigma_{route}(rt', dip)$ ))"

abbreviation

rt\_fresher\_syn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  rt  $\Rightarrow$  bool" ("(\_/  $\sqsubseteq$  \_)" [51, 999, 51] 50)

where

"rt1  $\sqsubseteq_i$  rt2  $\equiv$  rt\_fresher i rt1 rt2"

lemma rt\_fresher\_def':

"(rt1  $\sqsubseteq_i$  rt2) = (nsqn<sub>r</sub> (the (rt1 i)) < nsqn<sub>r</sub> (the (rt2 i))  $\vee$   
 nsqn<sub>r</sub> (the (rt1 i)) = nsqn<sub>r</sub> (the (rt2 i))  $\wedge$   $\pi_5$  (the (rt2 i))  $\leq$   $\pi_5$  (the (rt1 i)))"  
 unfolding rt\_fresher\_def fresher\_def by (rule refl)

lemma single\_rt\_fresher [intro]:

assumes "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"  
 shows "rt1  $\sqsubseteq_{ip}$  rt2"  
 using assms unfolding rt\_fresher\_def .

lemma rt\_fresher\_single [intro]:

assumes "rt1  $\sqsubseteq_{ip}$  rt2"  
 shows "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"  
 using assms unfolding rt\_fresher\_def .

lemma rt\_fresher\_def2:

assumes "dip  $\in$  kD(rt1)"  
 and "dip  $\in$  kD(rt2)"  
 shows "(rt1  $\sqsubseteq_{dip}$  rt2) = (nsqn rt1 dip < nsqn rt2 dip  
 $\vee$  (nsqn rt1 dip = nsqn rt2 dip  
 $\wedge$  the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip)))"  
 using assms unfolding rt\_fresher\_def fresher\_def by (simp add: kD\_nsqn proj5\_eq\_dhops)

lemma rt\_fresherI1 [intro]:

assumes "dip  $\in$  kD(rt1)"  
 and "dip  $\in$  kD(rt2)"  
 and "nsqn rt1 dip < nsqn rt2 dip"  
 shows "rt1  $\sqsubseteq_{dip}$  rt2"  
 unfolding rt\_fresher\_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt\_fresherI2 [intro]:

assumes "dip  $\in$  kD(rt1)"  
 and "dip  $\in$  kD(rt2)"  
 and "nsqn rt1 dip = nsqn rt2 dip"  
 and "the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip)"  
 shows "rt1  $\sqsubseteq_{dip}$  rt2"  
 unfolding rt\_fresher\_def2 [OF assms(1-2)] using assms(3-4) by simp

lemma rt\_fresherE [elim]:

```

assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  and "dip  $\in kD(rt1)$ "
  and "dip  $\in kD(rt2)$ "
  and "[[ nsqn rt1 dip < nsqn rt2 dip ]  $\implies P$  rt1 rt2 dip]"
  and "[[ nsqn rt1 dip = nsqn rt2 dip;
    the (dhops rt1 dip)  $\geq$  the (dhops rt2 dip) ]  $\implies P$  rt1 rt2 dip]"
shows "P rt1 rt2 dip"
using assms(1) unfolding rt_fresher_def2 [OF assms(2-3)]
using assms(4-5) by auto

lemma rt_fresher_refl [simp]: "rt  $\sqsubseteq_{dip}$  rt"
  unfolding rt_fresher_def by simp

lemma rt_fresher_trans [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
  using assms unfolding rt_fresher_def by auto

lemma rt_fresher_if_Some [intro!]:
  assumes "the (rt dip)  $\sqsubseteq r$ "
  shows "rt  $\sqsubseteq_{dip}$  ( $\lambda ip.$  if ip = dip then Some r else rt ip)"
  using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: "ip  $\Rightarrow$  rt  $\Rightarrow$  rt  $\Rightarrow$  bool"
where
  "rt_fresh_as  $\equiv \lambda dip$  rt1 rt2. (rt1  $\sqsubseteq_{dip}$  rt2)  $\wedge$  (rt2  $\sqsubseteq_{dip}$  rt1)"

abbreviation
  rt_fresh_as_syn :: "rt  $\Rightarrow$  ip  $\Rightarrow$  rt  $\Rightarrow$  bool" ("(_/  $\approx_{dip}$  _)" [51, 999, 51] 50)
where
  "rt1  $\approx_i$  rt2  $\equiv$  rt_fresh_as i rt1 rt2"

lemma rt_fresh_as_refl [simp]: " $\wedge$ rt dip. rt  $\approx_{dip}$  rt"
  unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
  " $\wedge$ rt1 rt2 rt3 dip. [[ rt1  $\approx_{dip}$  rt2; rt2  $\approx_{dip}$  rt3 ]  $\implies$  rt1  $\approx_{dip}$  rt3"
  unfolding rt_fresh_as_def rt_fresher_def
  by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt1"
  shows "rt1  $\approx_{dip}$  rt2"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
  assumes "dip  $\in kD(rt1)$ "
    and "dip  $\in kD(rt2)$ "
    and "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"
    and "the (rt2 dip)  $\sqsubseteq$  the (rt1 dip)"
  shows "rt1  $\approx_{dip}$  rt2"
  using assms unfolding rt_fresh_as_def
  by (clarsimp dest!: single_rt_fresher)

lemma nsqn_rt_fresh_asI:
  assumes "dip  $\in kD(rt)$ "
    and "dip  $\in kD(rt')$ "
    and "nsqn rt dip = nsqn rt' dip"
    and " $\pi_5$ (the (rt dip)) =  $\pi_5$ (the (rt' dip))"
  shows "rt  $\approx_{dip}$  rt'"
proof
  from assms(1-2,4) have dhops': "the (dhops rt' dip)  $\leq$  the (dhops rt dip)"

```

```

    by (simp add: proj5_eq_dhops)
  with assms(1-3) show "rt  $\sqsubseteq_{dip}$  rt'"
    by (rule rt_fresherI2)
next
  from assms(1-2,4) have dhops: "the (dhops rt dip)  $\leq$  the (dhops rt' dip)"
    by (simp add: proj5_eq_dhops)
  with assms(2,1) assms(3) [symmetric] show "rt'  $\sqsubseteq_{dip}$  rt"
    by (rule rt_fresherI2)
qed

lemma rt_fresh_asE [elim]:
  assumes "rt1  $\approx_{dip}$  rt2"
    and "[[ rt1  $\sqsubseteq_{dip}$  rt2; rt2  $\sqsubseteq_{dip}$  rt1 ]  $\implies$  P rt1 rt2 dip]"
  shows "P rt1 rt2 dip"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD1 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\sqsubseteq_{dip}$  rt1"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
  assumes "rt1  $\approx_{dip}$  rt2"
  shows "rt2  $\approx_{dip}$  rt1"
  using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
  assumes " $\neg$  (rt1  $\sqsubseteq_{dip}$  rt2)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
  proof
    assume "rt1  $\approx_{dip}$  rt2"
    hence "rt1  $\sqsubseteq_{dip}$  rt2" ..
    with ( $\neg$  (rt1  $\sqsubseteq_{dip}$  rt2)) show False ..
  qed

lemma not_rt_fresh_asI2 [intro]:
  assumes " $\neg$  (rt2  $\sqsubseteq_{dip}$  rt1)"
  shows " $\neg$  (rt1  $\approx_{dip}$  rt2)"
  proof
    assume "rt1  $\approx_{dip}$  rt2"
    hence "rt2  $\sqsubseteq_{dip}$  rt1" ..
    with ( $\neg$  (rt2  $\sqsubseteq_{dip}$  rt1)) show False ..
  qed

lemma not_single_rt_fresher [elim]:
  assumes " $\neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))"
  shows " $\neg$ (rt1  $\sqsubseteq_{ip}$  rt2)"
  proof
    assume "rt1  $\sqsubseteq_{ip}$  rt2"
    hence "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)" ..
    with ( $\neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))) show False ..
  qed

lemmas not_single_rt_fresh_asI1 [intro] = not_rt_fresh_asI1 [OF not_single_rt_fresher]
lemmas not_single_rt_fresh_asI2 [intro] = not_rt_fresh_asI2 [OF not_single_rt_fresher]

lemma not_rt_fresher_single [elim]:
  assumes " $\neg$ (rt1  $\sqsubseteq_{ip}$  rt2)"
  shows " $\neg$ (the (rt1 ip)  $\sqsubseteq$  the (rt2 ip))"

```

```

proof
  assume "the (rt1 ip)  $\sqsubseteq$  the (rt2 ip)"
  hence "rt1  $\sqsubseteq_{ip}$  rt2" ..
  with  $\langle \neg(rt1 \sqsubseteq_{ip} rt2) \rangle$  show False ..
qed

lemma rt_fresh_as_nsqnr:
  assumes "dip  $\in$  kD(rt1)"
    and "dip  $\in$  kD(rt2)"
    and "rt1  $\approx_{dip}$  rt2"
  shows "nsqnr (the (rt2 dip)) = nsqnr (the (rt1 dip))"
using assms(3) unfolding rt_fresh_as_def
by (auto simp: rt_fresher_def2 [OF  $\langle dip \in kD(rt1) \rangle \langle dip \in kD(rt2) \rangle$ ]
  rt_fresher_def2 [OF  $\langle dip \in kD(rt2) \rangle \langle dip \in kD(rt1) \rangle$ ]
  kD_nsqn [OF  $\langle dip \in kD(rt1) \rangle$ ]
  kD_nsqn [OF  $\langle dip \in kD(rt2) \rangle$ ])

lemma rt_fresher_mapupd [intro!]:
  assumes "dip  $\in$  kD(rt)"
    and "the (rt dip)  $\sqsubseteq$  r"
  shows "rt  $\sqsubseteq_{dip}$  rt(dip  $\mapsto$  r)"
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_map_update_other [intro!]:
  assumes "dip  $\in$  kD(rt)"
    and "dip  $\neq$  ip"
  shows "rt  $\sqsubseteq_{dip}$  rt(ip  $\mapsto$  r)"
using assms unfolding rt_fresher_def by simp

lemma rt_fresher_update_other [simp]:
  assumes inkD: "dip  $\in$  kD(rt)"
    and "dip  $\neq$  ip"
  shows "rt  $\sqsubseteq_{dip}$  update rt ip r"
using assms unfolding update_def
by (clarsimp split: option.split) (fastforce)

theorem rt_fresher_update [simp]:
  assumes "dip  $\in$  kD(rt)"
    and "the (dhops rt dip)  $\geq$  1"
    and "update_arg_wf r"
  shows "rt  $\sqsubseteq_{dip}$  update rt ip r"
proof (cases "dip = ip")
  assume "dip  $\neq$  ip" with  $\langle dip \in kD(rt) \rangle$  show ?thesis
    by (rule rt_fresher_update_other)
next
  assume "dip = ip"

  from  $\langle dip \in kD(rt) \rangle$  obtain dsnn dskn fn hopsn nhipn
    where rtn [simp]: "the (rt dip) = (dsnn, dskn, fn, hopsn, nhipn)"
    by (metis prod_cases5)
  with  $\langle the (dhops rt dip) \geq 1 \rangle$  and  $\langle dip \in kD(rt) \rangle$  have "hopsn  $\geq$  1"
    by (metis proj5_eq_dhops projs(4))
  from  $\langle dip \in kD(rt) \rangle$  rtn have [simp]: "sqn rt dip = dsnn"
    and [simp]: "the (dhops rt dip) = hopsn"
    and [simp]: "the (flag rt dip) = fn"
    by (simp add: sqn_def proj5_eq_dhops [symmetric]
      proj4_eq_flag [symmetric])
  from  $\langle update\_arg\_wf\ r \rangle$  have "(dsnn, dskn, fn, hopsn, nhipn)
     $\sqsubseteq$  the ((update rt dip r) dip)"
  proof (rule wf_r_cases)
    fix nhip
    from  $\langle hops_n \geq 1 \rangle$  have "(dsnn, dskn, fn, hopsn, nhipn)
       $\sqsubseteq$  (dsnn, unk, val, Suc 0, nhip)"

```



```

    unfolding fresher_def sqn_def by (cases fn) auto
  thus "(dsnn, dskn, fn, hopsn, nhipn)
    ⊆ the (update rt dip (0, unk, val, Suc 0, nhip) dip)"
    using ⟨dip∈kD(rt)⟩ by - (rule update_cases_kD, simp_all)
next
fix dsn :: sqn and hops nhip
assume "0 < dsn"
show "(dsnn, dskn, fn, hopsn, nhipn)
  ⊆ the (update rt dip (dsn, kno, val, hops, nhip) dip)"
proof (rule update_cases_kD [OF _ ⟨dip∈kD(rt)⟩], simp_all add: ⟨0 < dsn⟩)
  assume "dsnn < dsn"
  thus "(dsnn, dskn, fn, hopsn, nhipn)
    ⊆ (dsn, kno, val, hops, nhip)"
    unfolding fresher_def by auto
next
  assume "dsnn = dsn"
  and "hops < hopsn"
  thus "(dsn, dskn, fn, hopsn, nhipn)
    ⊆ (dsn, kno, val, hops, nhip)"
    unfolding fresher_def nsqnr_def by simp
next
  assume "dsnn = dsn"
  with ⟨0 < dsn⟩
  show "(dsn, dskn, inv, hopsn, nhipn)
    ⊆ (dsn, kno, val, hops, nhip)"
    unfolding fresher_def by simp
qed
qed
hence "rt ⊆dip update rt dip r"
  by - (rule single_rt_fresher, simp)
with ⟨dip = ip⟩ show ?thesis by simp
qed

```

theorem *rt\_fresher\_invalidate* [*simp*]:

```

assumes "dip∈kD(rt)"
  and indests: "∀rip∈dom(dests). rip∈vD(rt) ∧ sqn rt rip < the (dests rip)"
shows "rt ⊆dip invalidate rt dests"
proof (cases "dip∈dom(dests)")
  assume "dip∉dom(dests)"
  thus ?thesis using ⟨dip∈kD(rt)⟩
  by - (rule single_rt_fresher, simp)
next
  assume "dip∈dom(dests)"
  moreover with indests have "dip∈vD(rt)"
    and "sqn rt dip < the (dests dip)"
  by auto
  ultimately show ?thesis
  unfolding invalidate_def sqn_def
  by - (rule single_rt_fresher, auto simp: fresher_def)
qed

```

lemma *nsqn<sub>r</sub>\_invalidate* [*simp*]:

```

assumes "dip∈kD(rt)"
  and "dip∈dom(dests)"
shows "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
using assms unfolding invalidate_def by auto

```

lemma *rt\_fresh\_as\_inc\_invalidate* [*simp*]:

```

assumes "dip∈kD(rt)"
  and "∀rip∈dom(dests). rip∈vD(rt) ∧ the (dests rip) = inc (sqn rt rip)"
shows "rt ≈dip invalidate rt dests"
proof (cases "dip∈dom(dests)")
  assume "dip∉dom(dests)"
  with ⟨dip∈kD(rt)⟩ have "dip∈kD(invalidate rt dests)"

```

```

    by simp
  with ⟨dip∈kD(rt)⟩ show ?thesis
    by rule (simp_all add: ⟨dip∉dom(dests)⟩)
next
  assume "dip∈dom(dests)"
  with assms(2) have "dip∈vD(rt)"
    and "the (dests dip) = inc (sqn rt dip)" by auto
  from ⟨dip∈vD(rt)⟩ have "dip∈kD(rt)" by simp
  moreover then have "dip∈kD(invalidate rt dests)" by simp
  ultimately show ?thesis
  proof (rule nsqn_rt_fresh_asI)
    from ⟨dip∈vD(rt)⟩ have "nsqn rt dip = sqn rt dip" by simp
    also have "sqn rt dip = nsqnr (the (invalidate rt dests dip))"
    proof -
      from ⟨dip∈kD(rt)⟩ have "nsqnr (the (invalidate rt dests dip)) = the (dests dip) - 1"
        using ⟨dip∈dom(dests)⟩ by (rule nsqnr_invalidate)
      with ⟨the (dests dip) = inc (sqn rt dip)⟩
        show "sqn rt dip = nsqnr (the (invalidate rt dests dip))" by simp
    qed
    also from ⟨dip∈kD(invalidate rt dests)⟩
      have "nsqnr (the (invalidate rt dests dip)) = nsqn (invalidate rt dests) dip"
        by (simp add: kD_nsqn)
    finally show "nsqn rt dip = nsqn (invalidate rt dests) dip" .
  qed simp
qed

```

lemmas `rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]`

#### 5.5.4 Strictly comparing routing tables

definition `rt_strictly_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"`

where

`"rt_strictly_fresher ≡ λdip rt1 rt2. (rt1 ⊆dip rt2) ∧ ¬(rt1 ≈dip rt2)"`

abbreviation

`rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊆ _)" [51, 999, 51] 50)`

where

`"rt1 ⊆i rt2 ≡ rt_strictly_fresher i rt1 rt2"`

lemma `rt_strictly_fresher_def''`:

`"rt1 ⊆i rt2 = ((rt1 ⊆i rt2) ∧ ¬(rt2 ⊆i rt1))"`  
 unfolding `rt_strictly_fresher_def` `rt_fresh_as_def` by auto

lemma `rt_strictly_fresherI'` [intro]:

assumes `"rt1 ⊆i rt2"`  
 and `"¬(rt2 ⊆i rt1)"`  
 shows `"rt1 ⊆i rt2"`  
 using `assms` unfolding `rt_strictly_fresher_def''` by simp

lemma `rt_strictly_fresherE'` [elim]:

assumes `"rt1 ⊆i rt2"`  
 and `"[| rt1 ⊆i rt2; ¬(rt2 ⊆i rt1) |] ⇒ P rt1 rt2 i"`  
 shows `"P rt1 rt2 i"`  
 using `assms` unfolding `rt_strictly_fresher_def''` by simp

lemma `rt_strictly_fresherI` [intro]:

assumes `"rt1 ⊆i rt2"`  
 and `"¬(rt1 ≈i rt2)"`  
 shows `"rt1 ⊆i rt2"`  
 unfolding `rt_strictly_fresher_def` using `assms` ..

lemmas `rt_strictly_fresher_singleI [elim] = rt_strictly_fresherI [OF single_rt_fresher]`

lemma `rt_strictly_fresherE` [elim]:

```

assumes "rt1  $\sqsubseteq_i$  rt2"
  and "[[ rt1  $\sqsubseteq_i$  rt2;  $\neg$ (rt1  $\approx_i$  rt2) ] ]  $\implies$  P rt1 rt2 i"
  shows "P rt1 rt2 i"
using assms(1) unfolding rt_strictly_fresher_def
by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':
"rt1  $\sqsubseteq_i$  rt2 =
  (nsqnr (the (rt1 i)) < nsqnr (the (rt2 i))
   $\vee$  (nsqnr (the (rt1 i)) = nsqnr (the (rt2 i))  $\wedge$   $\pi_5$ (the (rt1 i)) >  $\pi_5$ (the (rt2 i))))"
unfolding rt_strictly_fresher_def'' rt_fresher_def fresher_def by auto

lemma rt_strictly_fresher_fresherD [dest]:
assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  shows "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)"
using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_not_fresh_asD [dest]:
assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  shows " $\neg$  rt1  $\approx_{dip}$  rt2"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
using assms proof -
  from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  obtain "the (rt1 dip)  $\sqsubseteq$  the (rt2 dip)" by auto
  also from  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  obtain "the (rt2 dip)  $\sqsubseteq$  the (rt3 dip)" by auto
  finally have "the (rt1 dip)  $\sqsubseteq$  the (rt3 dip)" .

  moreover have " $\neg$  (rt1  $\approx_{dip}$  rt3)"
  proof -
    from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  obtain " $\neg$ (the (rt2 dip)  $\sqsubseteq$  the (rt1 dip))" by auto
    also from  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  obtain " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt2 dip))" by auto
    finally have " $\neg$ (the (rt3 dip)  $\sqsubseteq$  the (rt1 dip))" .
    thus ?thesis ..
  qed
  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: " $\neg$  (rt  $\sqsubseteq_{dip}$  rt)"
using assms unfolding rt_strictly_fresher_def
by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
assumes "rt1  $\sqsubseteq_{dip}$  rt2"
  and "rt2  $\sqsubseteq_{dip}$  rt3"
  shows "rt1  $\sqsubseteq_{dip}$  rt3"
proof -
  from  $\langle$ rt1  $\sqsubseteq_{dip}$  rt2 $\rangle$  have "rt1  $\sqsubseteq_{dip}$  rt2"
    and " $\neg$ (rt2  $\sqsubseteq_{dip}$  rt1)"
  unfolding rt_strictly_fresher_def'' by auto
  from this(1) and  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  have "rt1  $\sqsubseteq_{dip}$  rt3" ..

  moreover from  $\langle$  $\neg$ (rt2  $\sqsubseteq_{dip}$  rt1) $\rangle$  have " $\neg$ (rt3  $\sqsubseteq_{dip}$  rt1)"
  proof (rule contrapos_nn)
    assume "rt3  $\sqsubseteq_{dip}$  rt1"
    with  $\langle$ rt2  $\sqsubseteq_{dip}$  rt3 $\rangle$  show "rt2  $\sqsubseteq_{dip}$  rt1" ..
  qed

  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3"
  unfolding rt_strictly_fresher_def'' by auto
qed

```

```

lemma rt_fresher_trans_rt_strictly_fresher' [elim, trans]:
  assumes "rt1  $\sqsubseteq_{dip}$  rt2"
    and "rt2  $\sqsubseteq_{dip}$  rt3"
    shows "rt1  $\sqsubseteq_{dip}$  rt3"
proof -
  from  $\langle rt2 \sqsubseteq_{dip} rt3 \rangle$  have "rt2  $\sqsubseteq_{dip}$  rt3"
    and " $\neg(rt3 \sqsubseteq_{dip} rt2)$ "
    unfolding rt_strictly_fresher_def'' by auto
  from  $\langle rt1 \sqsubseteq_{dip} rt2 \rangle$  and this(1) have "rt1  $\sqsubseteq_{dip}$  rt3" ..

  moreover from  $\langle \neg(rt3 \sqsubseteq_{dip} rt2) \rangle$  have " $\neg(rt3 \sqsubseteq_{dip} rt1)$ "
  proof (rule contrapos_nn)
    assume "rt3  $\sqsubseteq_{dip}$  rt1"
    thus "rt3  $\sqsubseteq_{dip}$  rt2" using  $\langle rt1 \sqsubseteq_{dip} rt2 \rangle$  ..
  qed

  ultimately show "rt1  $\sqsubseteq_{dip}$  rt3"
    unfolding rt_strictly_fresher_def'' by auto
qed

```

```

lemma rt_fresher_imp_nsqn_le:
  assumes "rt1  $\sqsubseteq_{ip}$  rt2"
    and "ip  $\in kD$  rt1"
    and "ip  $\in kD$  rt2"
  shows "nsqn rt1 ip  $\leq$  nsqn rt2 ip"
using assms(1)
by (auto simp add: rt_fresher_def2 [OF assms(2-3)])

```

```

lemma rt_strictly_fresher_ltI [intro]:
  assumes "dip  $\in kD$ (rt1)"
    and "dip  $\in kD$ (rt2)"
    and "nsqn rt1 dip < nsqn rt2 dip"
  shows "rt1  $\sqsubseteq_{dip}$  rt2"
proof
  from assms show "rt1  $\sqsubseteq_{dip}$  rt2" ..
next
  show " $\neg(rt1 \approx_{dip} rt2)$ "
  proof
    assume "rt1  $\approx_{dip}$  rt2"
    hence "rt2  $\sqsubseteq_{dip}$  rt1" ..
    hence "nsqn rt2 dip  $\leq$  nsqn rt1 dip"
      using  $\langle dip \in kD(rt2) \rangle$   $\langle dip \in kD(rt1) \rangle$ 
      by (rule rt_fresher_imp_nsqn_le)
    with  $\langle nsqn rt1 dip < nsqn rt2 dip \rangle$  show "False"
      by simp
  qed
qed

```

```

lemma rt_strictly_fresher_eqI [intro]:
  assumes "i  $\in kD$ (rt1)"
    and "i  $\in kD$ (rt2)"
    and "nsqn rt1 i = nsqn rt2 i"
    and " $\pi_5$ (the (rt2 i)) <  $\pi_5$ (the (rt1 i))"
  shows "rt1  $\sqsubseteq_i$  rt2"
using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

```

```

lemma invalidate_rtsf_left [simp]:
  " $\bigwedge$ dests dip rt rt'. dests dip = None  $\implies$  (invalidate rt dests  $\sqsubseteq_{dip}$  rt') = (rt  $\sqsubseteq_{dip}$  rt')"
  unfolding invalidate_def rt_strictly_fresher_def'
  by (rule iffI) (auto split: option.split_asm)

```

```

lemma vD_invalidate_rt_strictly_fresher [simp]:
  assumes "dip  $\in vD$ (invalidate rt1 dests)"

```

```

shows "(invalidate rt1 dests  $\sqsubset_{dip}$  rt2) = (rt1  $\sqsubset_{dip}$  rt2)"
proof (cases "dip  $\in$  dom(dests)")
  assume "dip  $\in$  dom(dests)"
  hence "dip  $\notin$  vD(invalidate rt1 dests)"
    unfolding invalidate_def vD_def
  by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
with  $\langle$ dip  $\in$  vD(invalidate rt1 dests) $\rangle$  show ?thesis by simp
next
  assume "dip  $\notin$  dom(dests)"
  hence "dests dip = None" by auto
  moreover with  $\langle$ dip  $\in$  vD(invalidate rt1 dests) $\rangle$  have "dip  $\in$  vD(rt1)"
    unfolding invalidate_def vD_def
  by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)
ultimately show ?thesis
  unfolding invalidate_def rt_strictly_fresher_def' by auto
qed

```

```

lemma rt_strictly_fresher_update_other [elim!]:
  " $\wedge$ dip ip rt r rt'.  $\llbracket$  dip  $\neq$  ip; rt  $\sqsubset_{dip}$  rt'  $\rrbracket \implies$  update rt ip r  $\sqsubset_{dip}$  rt'"
  unfolding rt_strictly_fresher_def' by clarsimp

```

```

lemma lt_sqn_imp_update_strictly_fresher:
  assumes "dip  $\in$  vD (rt2 nhip)"
    and *: "osn < sqn (rt2 nhip) dip"
    and **: "rt  $\neq$  update rt dip (osn, kno, val, hops, nhip)"
  shows "update rt dip (osn, kno, val, hops, nhip)  $\sqsubset_{dip}$  rt2 nhip"
  unfolding rt_strictly_fresher_def'
  proof (rule disjI1)
    from ** have "nsqn (update rt dip (osn, kno, val, hops, nhip)) dip = osn"
      by (rule nsqn_update_changed_kno_val)
    with  $\langle$ dip  $\in$  vD(rt2 nhip) $\rangle$ 
      have "nsqnr (the (update rt dip (osn, kno, val, hops, nhip) dip)) = osn"
        by (simp add: kD_nsqn)
    also have "osn < sqn (rt2 nhip) dip" by (rule *)
    also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip dip))"
      unfolding nsqnr_def using  $\langle$ dip  $\in$  vD (rt2 nhip) $\rangle$ 
      by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
    finally show "nsqnr (the (update rt dip (osn, kno, val, hops, nhip) dip))
      < nsqnr (the (rt2 nhip dip))" .
  qed

```

```

lemma dhops_le_hops_imp_update_strictly_fresher:
  assumes "dip  $\in$  vD(rt2 nhip)"
    and sqn: "sqn (rt2 nhip) dip = osn"
    and hop: "the (dhops (rt2 nhip) dip)  $\leq$  hops"
    and **: "rt  $\neq$  update rt dip (osn, kno, val, Suc hops, nhip)"
  shows "update rt dip (osn, kno, val, Suc hops, nhip)  $\sqsubset_{dip}$  rt2 nhip"
  unfolding rt_strictly_fresher_def'
  proof (rule disjI2, rule conjI)
    from ** have "nsqn (update rt dip (osn, kno, val, Suc hops, nhip)) dip = osn"
      by (rule nsqn_update_changed_kno_val)
    with  $\langle$ dip  $\in$  vD(rt2 nhip) $\rangle$ 
      have "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip) dip)) = osn"
        by (simp add: kD_nsqn)
    also have "osn = sqn (rt2 nhip) dip" by (rule sqn [symmetric])
    also have "sqn (rt2 nhip) dip = nsqnr (the (rt2 nhip dip))"
      unfolding nsqnr_def using  $\langle$ dip  $\in$  vD(rt2 nhip) $\rangle$ 
      by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
    finally show "nsqnr (the (update rt dip (osn, kno, val, Suc hops, nhip) dip))
      = nsqnr (the (rt2 nhip dip))" .
  next
    have "the (dhops (rt2 nhip) dip)  $\leq$  hops" by (rule hop)
    also have "hops < hops + 1" by simp
    also have "hops + 1 = the (dhops (update rt dip (osn, kno, val, Suc hops, nhip)) dip)"

```

```

using ** by simp
finally have "the (dhops (rt2 nhip) dip)
  < the (dhops (update rt dip (osn, kno, val, Suc hops, nhip)) dip)" .
thus " $\pi_5$  (the (rt2 nhip dip)) <  $\pi_5$  (the (update rt dip (osn, kno, val, Suc hops, nhip) dip))"
using ⟨dip ∈ vD(rt2 nhip)⟩ by (simp add: proj5_eq_dhops)
qed

```

```

lemma nsqn_invalidate:
assumes "dip ∈ kD(rt)"
  and " $\forall ip \in \text{dom}(\text{dests}). ip \in vD(rt) \wedge \text{the}(\text{dests } ip) = \text{inc}(\text{sqn } rt \text{ } ip)$ "
shows "nsqn (invalidate rt dests) dip = nsqn rt dip"
proof -
  from ⟨dip ∈ kD(rt)⟩ have "dip ∈ kD(invalidate rt dests)" by simp

  from assms have "rt  $\approx_{dip}$  invalidate rt dests"
  by (rule rt_fresh_as_inc_invalidate)
with ⟨dip ∈ kD(rt)⟩ ⟨dip ∈ kD(invalidate rt dests)⟩ show ?thesis
  by (simp add: kD_nsqn del: invalidate_kD_inv)
  (erule(2) rt_fresh_as_nsqnr)
qed

end

```

## 5.6 Invariant proofs on individual processes

```

theory E_Seq_Invariants
imports "../.../AWN/Invariants" E_Aadv E_Aadv_Data E_Aadv_Predicates E_Fresher
begin

```

The proposition numbers are taken from the December 2013 version of the Fehnker et al technical report.

Proposition 7.2

```

lemma sequence_number_increases:
"paadv i  $\models_A$  onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \text{sn } \xi \leq \text{sn } \xi')$ "
by inv_ctors

```

```

lemma sequence_number_one_or_bigger:
"paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, \_). 1 \leq \text{sn } \xi)$ "
by (rule onll_step_to_invariantI [OF sequence_number_increases])
(auto simp:  $\sigma_{AODV\_def}$ )

```

We can get rid of the onl/onll if desired...

```

lemma sequence_number_increases':
"paadv i  $\models_A$  ( $\lambda((\xi, \_), \_, (\xi', \_)). \text{sn } \xi \leq \text{sn } \xi')$ "
by (rule step_invariant_weakenE [OF sequence_number_increases]) (auto dest!: onllD)

```

```

lemma sequence_number_one_or_bigger':
"paadv i  $\models$  ( $\lambda(\xi, \_). 1 \leq \text{sn } \xi)$ "
by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto

```

```

lemma sip_in_kD:
"paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, 1). 1 \in (\{PAadv-:7\} \cup \{PAadv-:5\} \cup \{PRrep-:0..PRrep-:4\}
  \cup \{PRreq-:0..PRreq-:3\}) \longrightarrow \text{sip } \xi \in kD(rt \ \xi))"$ 
by inv_ctors

```

Proposition 7.38

```

lemma includes_nhip:
"paadv i  $\models$  onl  $\Gamma_{AODV} (\lambda(\xi, 1). \forall dip \in kD(rt \ \xi). \text{the}(\text{nhop}(rt \ \xi) \ dip) \in kD(rt \ \xi))"$ 
proof -
  { fix ip and  $\xi \ \xi' :: \text{state}$ 
  assume " $\forall dip \in kD(rt \ \xi). \text{the}(\text{nhop}(rt \ \xi) \ dip) \in kD(rt \ \xi)$ "
  and " $\xi' = \xi(|rt := \text{update}(rt \ \xi) \ ip \ 0, \text{unk}, \text{val}, \text{Suc } 0, ip)$ "
  hence " $\forall dip \in kD(rt \ \xi).$ "

```

```

      the (nhop (update (rt  $\xi$ ) ip (0, unk, val, Suc 0, ip)) dip) = ip
    ∨ the (nhop (update (rt  $\xi$ ) ip (0, unk, val, Suc 0, ip)) dip) ∈ kD (rt  $\xi$ )"
  by clarsimp (metis nhop_update_unk_val update_another)
} note one_hop = this
{
  fix ip sip sn hops and  $\xi \xi' :: state$ 
  assume "∀dip∈kD (rt  $\xi$ ). the (nhop (rt  $\xi$ ) dip) ∈ kD (rt  $\xi$ )"
  and " $\xi' = \xi(\text{rt} := \text{update (rt } \xi) \text{ ip (sn, kno, val, Suc hops, sip)})$ "
  and " $\text{sip} \in \text{kD (rt } \xi)$ "
  hence "(the (nhop (update (rt  $\xi$ ) ip (sn, kno, val, Suc hops, sip)) ip) = ip
    ∨ the (nhop (update (rt  $\xi$ ) ip (sn, kno, val, Suc hops, sip)) ip) ∈ kD (rt  $\xi$ ))
    ∧ (∀dip∈kD (rt  $\xi$ ).
      the (nhop (update (rt  $\xi$ ) ip (sn, kno, val, Suc hops, sip)) dip) = ip
      ∨ the (nhop (update (rt  $\xi$ ) ip (sn, kno, val, Suc hops, sip)) dip) ∈ kD (rt  $\xi$ ))"
  by (metis kD_update_unchanged nhop_update_changed update_another)
} note nhop_is_sip = this
show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf sip_in_kD]
      solve: one_hop nhop_is_sip)
qed

```

Proposition 7.4

lemma known\_destinations\_increase:

```

"paadv i ⊨A onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \text{kD (rt } \xi) \subseteq \text{kD (rt } \xi')$ )"
by (inv_cterms simp add: subset_insertI)

```

Proposition 7.5

lemma rreqs\_increase:

```

"paadv i ⊨A onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \text{rreqs } \xi \subseteq \text{rreqs } \xi')$ "
by (inv_cterms simp add: subset_insertI)

```

lemma dests\_bigger\_than\_sqn:

```

"paadv i ⊨ onl  $\Gamma_{AODV} (\lambda(\xi, l). l \in \{\text{PAadv-:15..PAadv-:17}\}
  \cup \{\text{PPkt-:7..PPkt-:9}\}
  \cup \{\text{PRreq-:11..PRreq-:13}\}
  \cup \{\text{PRreq-:20..PRreq-:22}\}
  \cup \{\text{PRrep-:7..PRrep-:9}\}
  \cup \{\text{PRerr-:1..PRerr-:4}\} \cup \{\text{PRerr-:6}\}
  \rightarrow (\forall \text{ip} \in \text{dom}(\text{dests } \xi). \text{ip} \in \text{kD}(\text{rt } \xi) \wedge \text{sqn (rt } \xi) \text{ ip} \leq \text{the (dests } \xi \text{ ip)}))"$ 

```

proof -

have sqninv:

```

"∧dests rt rsn ip.
  [ ∃ip∈dom(dests). ip∈kD(rt) ∧ sqn rt ip ≤ the (dests ip); dests ip = Some rsn ]
  ⇒ sqn (invalidate rt dests) ip ≤ rsn"
  by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto

```

have indests:

```

"∧dests rt rsn ip.
  [ ∃ip∈dom(dests). ip∈kD(rt) ∧ sqn rt ip ≤ the (dests ip); dests ip = Some rsn ]
  ⇒ ip∈kD(rt) ∧ sqn rt ip ≤ rsn"
  by (metis domI option.sel)

```

show ?thesis

```

  by inv_cterms
     (clarsimp split: split_if_asm option.split_asm
      elim!: sqninv indests)+

```

qed

Proposition 7.6

lemma sqns\_increase:

```

"paadv i ⊨A onll  $\Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \forall \text{ip}. \text{sqn (rt } \xi) \text{ ip} \leq \text{sqn (rt } \xi') \text{ ip})"$ 

```

proof -

```

{
  fix  $\xi :: state$ 
  assume *: "∀ip∈dom(dests  $\xi$ ). ip ∈ kD (rt  $\xi$ ) ∧ sqn (rt  $\xi$ ) ip ≤ the (dests  $\xi$  ip)"
  have "∀ip. sqn (rt  $\xi$ ) ip ≤ sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) ip"
  proof
    fix ip

```

```

from * have "ip $\notin$ dom(dests  $\xi$ )  $\vee$  sqn (rt  $\xi$ ) ip  $\leq$  the (dests  $\xi$  ip)" by simp
thus "sqn (rt  $\xi$ ) ip  $\leq$  sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) ip"
  by (metis domI invalidate_sqn option.sel)

```

qed

```

} note solve_invalidate = this

```

```

show ?thesis

```

```

  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf dests_bigger_than_sqn]
      simp add: solve_invalidate)

```

qed

Proposition 7.7

lemma ip\_constant:

```

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_).$  ip  $\xi = i$ )"
by (inv_cterms simp add:  $\sigma_{AODV\_def}$ )

```

Proposition 7.8

lemma sender\_ip\_valid':

```

"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_).$  anycast ( $\lambda m.$  not_Pkt m  $\longrightarrow$  msg_sender m = ip  $\xi$ ) a)"
by inv_cterms

```

lemma sender\_ip\_valid:

```

"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_).$  anycast ( $\lambda m.$  not_Pkt m  $\longrightarrow$  msg_sender m = i) a)"
by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid'])
(auto dest!: onlD onllD)

```

lemma received\_msg\_inv:

```

"paadv i  $\models$  (recvmmsg P  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l).$  l  $\in$  {PAadv-:1}  $\longrightarrow$  P (msg  $\xi$ ))"
by inv_cterms

```

Proposition 7.9

lemma sip\_not\_ip':

```

"paadv i  $\models$  (recvmmsg ( $\lambda m.$  not_Pkt m  $\longrightarrow$  msg_sender m  $\neq$  i)  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_).$  sip  $\xi \neq$  ip  $\xi$ )"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]
    onl_invariant_sterms [OF aadv_wf ip_constant [THEN invariant_restrict_inD]]
    simp add: clear_locals_sip_not_ip') clarsimp+

```

lemma sip\_not\_ip:

```

"paadv i  $\models$  (recvmmsg ( $\lambda m.$  not_Pkt m  $\longrightarrow$  msg_sender m  $\neq$  i)  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_).$  sip  $\xi \neq$  i)"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]
    onl_invariant_sterms [OF aadv_wf ip_constant [THEN invariant_restrict_inD]]
    simp add: clear_locals_sip_not_ip') clarsimp+

```

Neither sip\_not\_ip' nor sip\_not\_ip is needed to show loop freedom.

Proposition 7.10

lemma hop\_count\_positive:

```

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, \_).$   $\forall ip \in kD$  (rt  $\xi$ ). the (dhops (rt  $\xi$ ) ip)  $\geq$  1)"
by (inv_cterms) auto

```

lemma rreq\_dip\_in\_vD\_dip\_eq\_ip:

```

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l).$  (l  $\in$  {PRreq-:16..PRreq-:17}  $\longrightarrow$  dip  $\xi \in$  vD(rt  $\xi$ ))
   $\wedge$  (l  $\in$  {PRreq-:6, PRreq-:7}  $\longrightarrow$  dip  $\xi =$  ip  $\xi$ )
   $\wedge$  (l  $\in$  {PRreq-:15..PRreq-:17}  $\longrightarrow$  dip  $\xi \neq$  ip  $\xi$ ))"

```

by inv\_cterms

lemma rrep\_dip\_in\_vD:

```

"paadv i  $\models$  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l).$  (l  $\in$  {PRrep-:4}  $\longrightarrow$  dip  $\xi \in$  vD(rt  $\xi$ )))"
by inv_cterms

```

Proposition 7.11

lemma anycast\_msg\_zhops:

```

" $\bigwedge$ rreqid dip dsn dsk oip osn sip.
  paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_).$  anycast msg_zhops a)"

```



```

proof (inv_cterms inv add:
  onl_invariant_sterms [OF aadv_wf rreq_dip_in_vD_dip_eq_ip]
  onl_invariant_sterms [OF aadv_wf rrep_dip_in_vD]
  onl_invariant_sterms [OF aadv_wf hop_count_positive],
  elim conjE)
fix l  $\xi$  a pp p' pp'
assume "( $\xi$ , pp)  $\in$  reachable (paadv i) TT"
  and "{PRreq-:16}unicast( $\lambda\xi$ . the (nhop (rt  $\xi$ ) (oip  $\xi$ )),
     $\lambda\xi$ . Rrep (the (dhops (rt  $\xi$ ) (dip  $\xi$ ))) (dip  $\xi$ ) (sqn (rt  $\xi$ ) (dip  $\xi$ )) (oip  $\xi$ ) (ip  $\xi$ )).
    p'  $\triangleright$  pp'  $\in$  sterms  $\Gamma_{AODV}$  pp"
  and "l = PRreq-:16"
  and "a = unicast (the (nhop (rt  $\xi$ ) (oip  $\xi$ )))
    (Rrep (the (dhops (rt  $\xi$ ) (dip  $\xi$ ))) (dip  $\xi$ ) (sqn (rt  $\xi$ ) (dip  $\xi$ )) (oip  $\xi$ ) (ip  $\xi$ ))"
  and *: " $\forall ip \in kD$  (rt  $\xi$ ). Suc 0  $\leq$  the (dhops (rt  $\xi$ ) ip)"
  and "dip  $\xi \in vD$  (rt  $\xi$ )"
from (dip  $\xi \in vD$  (rt  $\xi$ )) have "dip  $\xi \in kD$  (rt  $\xi$ )"
  by (rule vD_ID_gives_kD(1))
with * have "Suc 0  $\leq$  the (dhops (rt  $\xi$ ) (dip  $\xi$ ))" ..
thus "0 < the (dhops (rt  $\xi$ ) (dip  $\xi$ ))" by simp
next
fix l  $\xi$  a pp p' pp'
assume "( $\xi$ , pp)  $\in$  reachable (paadv i) TT"
  and "{PRrep-:4}unicast( $\lambda\xi$ . the (nhop (rt  $\xi$ ) (oip  $\xi$ )),
     $\lambda\xi$ . Rrep (the (dhops (rt  $\xi$ ) (dip  $\xi$ ))) (dip  $\xi$ ) (sqn (rt  $\xi$ ) (dip  $\xi$ )) (oip  $\xi$ ) (ip  $\xi$ )).
    p'  $\triangleright$  pp'  $\in$  sterms  $\Gamma_{AODV}$  pp"
  and "l = PRrep-:4"
  and "a = unicast (the (nhop (rt  $\xi$ ) (oip  $\xi$ )))
    (Rrep (the (dhops (rt  $\xi$ ) (dip  $\xi$ ))) (dip  $\xi$ ) (sqn (rt  $\xi$ ) (dip  $\xi$ )) (oip  $\xi$ ) (ip  $\xi$ ))"
  and *: " $\forall ip \in kD$  (rt  $\xi$ ). Suc 0  $\leq$  the (dhops (rt  $\xi$ ) ip)"
  and "dip  $\xi \in vD$  (rt  $\xi$ )"
from (dip  $\xi \in vD$  (rt  $\xi$ )) have "dip  $\xi \in kD$  (rt  $\xi$ )"
  by (rule vD_ID_gives_kD(1))
with * have "Suc 0  $\leq$  the (dhops (rt  $\xi$ ) (dip  $\xi$ ))" ..
thus "the (dhops (rt  $\xi$ ) (dip  $\xi$ )) = 0  $\longrightarrow$  dip  $\xi$  = ip  $\xi$ "
  by auto
qed

```

lemma hop\_count\_zero\_oip\_dip\_sip:

```

"paadv i  $\models$  (recvmsg msg_zhops  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .
  (l  $\in$  {PAadv-:4..PAadv-:5}  $\cup$  {PRreq-:n/n. True}  $\longrightarrow$ 
    (hops  $\xi$  = 0  $\longrightarrow$  oip  $\xi$  = sip  $\xi$ ))
   $\wedge$ 
  ((l  $\in$  {PAadv-:6..PAadv-:7}  $\cup$  {PRrep-:n/n. True}  $\longrightarrow$ 
    (hops  $\xi$  = 0  $\longrightarrow$  dip  $\xi$  = sip  $\xi$ )))"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]) auto

```

lemma osn\_rreq:

```

"paadv i  $\models$  (recvmsg rreq_rrep_sn  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .
  l  $\in$  {PAadv-:4, PAadv-:5}  $\cup$  {PRreq-:n/n. True}  $\longrightarrow$  l  $\leq$  osn  $\xi$ )"
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf received_msg_inv]) clarsimp

```

lemma osn\_rreq':

```

"paadv i  $\models$  (recvmsg ( $\lambda m$ . rreq_rrep_sn m  $\wedge$  msg_zhops m)  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .
  l  $\in$  {PAadv-:4, PAadv-:5}  $\cup$  {PRreq-:n/n. True}  $\longrightarrow$  l  $\leq$  osn  $\xi$ )"
proof (rule invariant_weakenE [OF osn_rreq])
  fix a
  assume "recvmsg ( $\lambda m$ . rreq_rrep_sn m  $\wedge$  msg_zhops m) a"
  thus "recvmsg rreq_rrep_sn a"
    by (cases a) simp_all
qed

```

lemma dsn\_rrep:

```

"paadv i  $\models$  (recvmsg rreq_rrep_sn  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ .
  l  $\in$  {PAadv-:6, PAadv-:7}  $\cup$  {PRrep-:n/n. True}  $\longrightarrow$  l  $\leq$  dsn  $\xi$ )"

```

by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]) clarsimp

lemma dsn\_rrep':

```
"paadv i  $\models$  (recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l).$ 
 $l \in \{PAadv-:6, PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow 1 \leq dsn\ \xi$ )"
proof (rule invariant_weakenE [OF dsn_rrep])
  fix a
  assume "recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ ) a"
  thus "recvmsg rreq_rrep_sn a"
    by (cases a) simp_all
qed
```

lemma hop\_count\_zero\_oip\_dip\_sip':

```
"paadv i  $\models$  (recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l).$ 
 $(l \in \{PAadv-:4..PAadv-:5\} \cup \{PRreq-:n/n. True\} \rightarrow$ 
 $(hops\ \xi = 0 \rightarrow oip\ \xi = sip\ \xi))$ 
 $\wedge$ 
 $((l \in \{PAadv-:6..PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow$ 
 $(hops\ \xi = 0 \rightarrow dip\ \xi = sip\ \xi))))$ "
proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])
  fix a
  assume "recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ ) a"
  thus "recvmsg msg_zhops a"
    by (cases a) simp_all
qed
```

Proposition 7.12

lemma zero\_seq\_unk\_hops\_one':

```
"paadv i  $\models$  (recvmsg ( $\lambda m. rreq\_rrep\_sn\ m \wedge msg\_zhops\ m$ )  $\rightarrow$ ) onl  $\Gamma_{AODV}$  ( $\lambda(\xi, _).$ 
 $\forall dip \in kD(rt\ \xi). (sqn\ (rt\ \xi)\ dip = 0 \rightarrow sqnf\ (rt\ \xi)\ dip = unk)$ 
 $\wedge (sqnf\ (rt\ \xi)\ dip = unk \rightarrow the\ (dhops\ (rt\ \xi)\ dip) = 1)$ 
 $\wedge (the\ (dhops\ (rt\ \xi)\ dip) = 1 \rightarrow the\ (nhop\ (rt\ \xi)\ dip) = dip))$ "
proof -
{ fix dip and  $\xi :: state$  and P
  assume "sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip = 0"
    and all: " $\forall ip. sqn\ (rt\ \xi)\ ip \leq sqn\ (invalidate\ (rt\ \xi)\ (dests\ \xi))\ ip$ "
    and *: "sqn (rt  $\xi$ ) dip = 0  $\implies$  P  $\xi$  dip"
  have "P  $\xi$  dip"
  proof -
    from all have "sqn (rt  $\xi$ ) dip  $\leq$  sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip" ..
    with (sqn (invalidate (rt  $\xi$ ) (dests  $\xi$ )) dip = 0) have "sqn (rt  $\xi$ ) dip = 0" by simp
    thus "P  $\xi$  dip" by (rule *)
  qed
} note sqn_invalidate_zero [elim!] = this
```

```
{ fix dsn hops :: nat and sip oip rt and ip dip :: ip
  assume " $\forall dip \in kD(rt).$ 
 $(sqn\ rt\ dip = 0 \rightarrow \pi_3(the\ (rt\ dip)) = unk) \wedge$ 
 $(\pi_3(the\ (rt\ dip)) = unk \rightarrow the\ (dhops\ rt\ dip) = Suc\ 0) \wedge$ 
 $(the\ (dhops\ rt\ dip) = Suc\ 0 \rightarrow the\ (nhop\ rt\ dip) = dip)$ "
  and "hops = 0  $\rightarrow$  sip = dip"
  and "Suc 0  $\leq$  dsn"
  and "ip  $\neq$  dip  $\rightarrow ip \in kD(rt)"
  hence "the (dhops (update rt dip (dsn, kno, val, Suc hops, sip)) ip) = Suc 0  $\rightarrow$ 
  the (nhop (update rt dip (dsn, kno, val, Suc hops, sip)) ip) = ip"
  by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
} note prreq_ok1 [simp] = this$ 
```

```
{ fix ip dsn hops sip oip rt dip
  assume " $\forall dip \in kD(rt).$ 
 $(sqn\ rt\ dip = 0 \rightarrow \pi_3(the\ (rt\ dip)) = unk) \wedge$ 
 $(\pi_3(the\ (rt\ dip)) = unk \rightarrow the\ (dhops\ rt\ dip) = Suc\ 0) \wedge$ 
 $(the\ (dhops\ rt\ dip) = Suc\ 0 \rightarrow the\ (nhop\ rt\ dip) = dip)$ "
  and "Suc 0  $\leq$  dsn"
```

```

    and "ip ≠ dip → ip ∈ kD(rt)"
  hence "π3(the (update rt dip (dsn, kno, val, Suc hops, sip) ip)) = unk →
    the (dhops (update rt dip (dsn, kno, val, Suc hops, sip) ip) = Suc 0)"
    by - (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)
} note prreq_ok2 [simp] = this

{ fix ip dsn hops sip oip rt dip
  assume "∀ dip ∈ kD(rt).
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
    and "Suc 0 ≤ dsn"
    and "ip ≠ dip → ip ∈ kD(rt)"
  hence "sqn (update rt dip (dsn, kno, val, Suc hops, sip) ip) = 0 →
    π3 (the (update rt dip (dsn, kno, val, Suc hops, sip) ip)) = unk"
    by - (rule update_cases, auto simp add: sqn_def dest!: bspec)
} note prreq_ok3 [simp] = this

{ fix rt sip
  assume "∀ dip ∈ kD rt.
    (sqn rt dip = 0 → π3(the (rt dip)) = unk) ∧
    (π3(the (rt dip)) = unk → the (dhops rt dip) = Suc 0) ∧
    (the (dhops rt dip) = Suc 0 → the (nhop rt dip) = dip)"
  hence "∀ dip ∈ kD rt.
    (sqn (update rt sip (0, unk, val, Suc 0, sip) dip) = 0 →
    π3(the (update rt sip (0, unk, val, Suc 0, sip) dip)) = unk)
    ∧ (π3(the (update rt sip (0, unk, val, Suc 0, sip) dip)) = unk →
    the (dhops (update rt sip (0, unk, val, Suc 0, sip) dip) = Suc 0)
    ∧ (the (dhops (update rt sip (0, unk, val, Suc 0, sip) dip) = Suc 0 →
    the (nhop (update rt sip (0, unk, val, Suc 0, sip) dip) = dip)"
    by - (rule update_cases, simp_all add: sqn_def sqnf_def)
} note prreq_ok4 [simp] = this

have prreq_ok5 [simp]: "∧ sip rt.
  π3(the (update rt sip (0, unk, val, Suc 0, sip) sip)) = unk →
  the (dhops (update rt sip (0, unk, val, Suc 0, sip) sip) = Suc 0"
  by (rule update_cases) simp_all

have prreq_ok6 [simp]: "∧ sip rt.
  sqn (update rt sip (0, unk, val, Suc 0, sip) sip) = 0 →
  π3 (the (update rt sip (0, unk, val, Suc 0, sip) sip)) = unk"
  by (rule update_cases) simp_all

show ?thesis
  by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf hop_count_zero_oip_dip_sip']
    seq_step_invariant_sterms_TT [OF sqns_increase aadv_wf aadv_trans]
    onl_invariant_sterms [OF aadv_wf osn_rreq']
    onl_invariant_sterms [OF aadv_wf dsn_rrep']) clarsimp+
qed

lemma zero_seq_unk_hops_one:
  "paadv i ⊨ (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl ΓAODV (λ(ξ, _).
    ∀ dip ∈ kD(rt ξ). (sqn (rt ξ) dip = 0 → (sqnf (rt ξ) dip = unk
      ∧ the (dhops (rt ξ) dip) = 1
      ∧ the (nhop (rt ξ) dip) = dip)))"
  by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

lemma kD_unk_or_atleast_one:
  "paadv i ⊨ (recvmsg rreq_rrep_sn →) onl ΓAODV (λ(ξ, l).
    ∀ dip ∈ kD(rt ξ). π3(the (rt ξ dip)) = unk ∨ 1 ≤ π2(the (rt ξ dip)))"
  proof -
    { fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2
      assume "dsk1 = unk ∨ Suc 0 ≤ dsn2"
      hence "π3(the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1) sip)) = unk"
    }
  }

```

```

    ∨ Suc 0 ≤ sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2)) sip"
  unfolding update_def by (cases "dsk1 =unk") (clarsimp split: option.split)+
} note fromsip [simp] = this

{ fix dip sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2
  assume allkd: "∀dip∈kD(rt). π3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip"
  and **: "dsk1 = unk ∨ Suc 0 ≤ dsn2"
  have "∀dip∈kD(rt). π3(the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1) dip)) = unk
    ∨ Suc 0 ≤ sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2)) dip"
    (is "∀dip∈kD(rt). ?prop dip")
  proof
    fix dip
    assume "dip∈kD(rt)"
    thus "?prop dip"
    proof (cases "dip = sip")
      assume "dip = sip"
      with ** show ?thesis
      by simp
    next
      assume "dip ≠ sip"
      with ⟨dip∈kD(rt)⟩ allkd show ?thesis
      by simp
    qed
  qed
} note solve_update [simp] = this

{ fix dip rt dests
  assume *: "∀ip∈dom(dests). ip∈kD(rt) ∧ sqn rt ip ≤ the (dests ip)"
  and **: "∀ip∈kD(rt). π3(the (rt ip)) = unk ∨ Suc 0 ≤ sqn rt ip"
  have "∀dip∈kD(rt). π3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"
  proof
    fix dip
    assume "dip∈kD(rt)"
    with ** have "π3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip" ..
    thus "π3 (the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests) dip"
    proof
      assume "π3(the (rt dip)) = unk" thus ?thesis ..
    next
      assume "Suc 0 ≤ sqn rt dip"
      have "Suc 0 ≤ sqn (invalidate rt dests) dip"
      proof (cases "dip∈dom(dests)")
        assume "dip∈dom(dests)"
        with * have "sqn rt dip ≤ the (dests dip)" by simp
        with ⟨Suc 0 ≤ sqn rt dip⟩ have "Suc 0 ≤ the (dests dip)" by simp
        with ⟨dip∈dom(dests)⟩ ⟨dip∈kD(rt)⟩ [THEN kD_Some] show ?thesis
        unfolding invalidate_def sqn_def by auto
      next
        assume "dip∉dom(dests)"
        with ⟨Suc 0 ≤ sqn rt dip⟩ ⟨dip∈kD(rt)⟩ [THEN kD_Some] show ?thesis
        unfolding invalidate_def sqn_def by auto
      qed
    qed
    thus ?thesis by (rule disjI2)
  qed
} note solve_invalidate [simp] = this

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn
                                             [THEN invariant_restrict_inD]]
    onl_invariant_sterms [OF aodv_wf osn_rreq]
    onl_invariant_sterms [OF aodv_wf dsn_rrep]
    simp add: proj3_inv proj2_eq_sqn)
qed

```

Proposition 7.13

lemma rreq\_rrep\_sn\_any\_step\_invariant:

"paadv i  $\models_A$  (recvmsg rreq\_rrep\_sn  $\rightarrow$ ) onll  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_)$ . anycast rreq\_rrep\_sn a)"

proof -

have sqnf\_kno: "paadv i  $\models$  onll  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ ).

( $1 \in \{\text{PRreq-:16}\} \rightarrow \text{dip } \xi \in \text{kD}(\text{rt } \xi) \wedge \text{sqnf}(\text{rt } \xi) (\text{dip } \xi) = \text{kno}$ )

by (inv\_cterms)

have rrep\_sqn\_greater\_dsn: "paadv i  $\models$  (recvmsg rreq\_rrep\_sn  $\rightarrow$ ) onll  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ ).

( $1 \in \{\text{PRrep-:1} \dots \text{PRrep-:4}\} \rightarrow 1 \leq \text{sqn}(\text{rt } \xi) (\text{dip } \xi)$ )"

by (inv\_cterms inv add: onll\_invariant\_sterms [OF aadv\_wf received\_msg\_inv]

onll\_invariant\_sterms [OF aadv\_wf dsn\_rrep])

(clarsimp simp: update\_kno\_dsn\_greater\_zero [simplified])

show ?thesis

by (inv\_cterms inv add: onll\_invariant\_sterms [OF aadv\_wf sequence\_number\_one\_or\_bigger  
[THEN invariant\_restrict\_inD]]

onll\_invariant\_sterms [OF aadv\_wf kD\_unk\_or\_atleast\_one]

onll\_invariant\_sterms\_TT [OF aadv\_wf sqnf\_kno]

onll\_invariant\_sterms [OF aadv\_wf osn\_rreq]

onll\_invariant\_sterms [OF aadv\_wf dsn\_rrep]

onll\_invariant\_sterms [OF aadv\_wf rrep\_sqn\_greater\_dsn])

(auto simp: proj2\_eq\_sqn)

qed

Proposition 7.14

lemma rreq\_rrep\_fresh\_any\_step\_invariant:

"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast (rreq\_rrep\_fresh (rt  $\xi$ )) a)"

proof -

have rreq\_oip: "paadv i  $\models$  onll  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ ).

( $1 \in \{\text{PRreq-:3} \dots \text{PRreq-:9}\} \cup \{\text{PRreq-:15}, \text{PRreq-:24}, \text{PRreq-:26}\}$

$\rightarrow \text{oip } \xi \in \text{kD}(\text{rt } \xi)$

$\wedge (\text{sqn}(\text{rt } \xi) (\text{oip } \xi) > (\text{osn } \xi)$

$\vee (\text{sqn}(\text{rt } \xi) (\text{oip } \xi) = (\text{osn } \xi)$

$\wedge \text{the}(\text{dhops}(\text{rt } \xi) (\text{oip } \xi)) \leq \text{Suc}(\text{hops } \xi)$

$\wedge \text{the}(\text{flag}(\text{rt } \xi) (\text{oip } \xi)) = \text{val}))$ "

proof inv\_cterms

fix l  $\xi$  l' pp p'

assume " $(\xi, \text{pp}) \in \text{reachable}(\text{paadv } i)$  TT"

and " $\{\text{PRreq-:2}\} \Vdash \lambda \xi. \xi(\text{rt} :=$

$\text{update}(\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops } \xi), \text{sip } \xi)) \Vdash p' \in \text{sterms } \Gamma_{AODV} \text{pp}$ "

and " $l' = \text{PRreq-:3}$ "

show " $\text{osn } \xi < \text{sqn}(\text{update}(\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops } \xi), \text{sip } \xi)) (\text{oip } \xi)$

$\vee (\text{sqn}(\text{update}(\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops } \xi), \text{sip } \xi)) (\text{oip } \xi) = \text{osn } \xi$

$\wedge \text{the}(\text{dhops}(\text{update}(\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops } \xi), \text{sip } \xi)) (\text{oip } \xi))$

$\leq \text{Suc}(\text{hops } \xi)$

$\wedge \text{the}(\text{flag}(\text{update}(\text{rt } \xi) (\text{oip } \xi) (\text{osn } \xi, \text{kno}, \text{val}, \text{Suc}(\text{hops } \xi), \text{sip } \xi)) (\text{oip } \xi))$

$= \text{val})$ "

unfolding update\_def by (clarsimp split: option.split)

(metis linorder\_neqE\_nat not\_less)

qed

have rrep\_prrep: "paadv i  $\models$  onll  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ ).

( $1 \in \{\text{PRrep-:4}\} \rightarrow (\text{dip } \xi \in \text{kD}(\text{rt } \xi)$

$\wedge \text{the}(\text{flag}(\text{rt } \xi) (\text{dip } \xi)) = \text{val}))$ "

by (inv\_cterms inv add: onll\_invariant\_sterms [OF aadv\_wf sip\_in\_kD])

have rreq\_oip\_kD: "paadv i  $\models$  onll  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ . ( $1 \in \{\text{PRreq-:3} \dots \text{PRreq-:22}\} \rightarrow \text{oip } \xi \in \text{kD}(\text{rt } \xi)$ ))"

by(inv\_cterms)

have rreq\_dip\_kD\_oip\_sqn: "paadv i  $\models$  onll  $\Gamma_{AODV}$  ( $\lambda(\xi, 1)$ ).

( $1 \in \{\text{PRreq-:16} \dots \text{PRreq-:17}\}$

$\rightarrow (\text{dip } \xi \in \text{kD}(\text{rt } \xi)$

$\wedge (\text{sqn}(\text{rt } \xi) (\text{oip } \xi) > (\text{osn } \xi)$

$$\begin{aligned} & \vee (\text{sqn } (rt \ \xi) \text{ (oip } \xi) = (\text{osn } \xi) \\ & \quad \wedge \text{the } (\text{dhops } (rt \ \xi) \text{ (oip } \xi)) \leq \text{Suc } (\text{hops } \xi) \\ & \quad \wedge \text{the } (\text{flag } (rt \ \xi) \text{ (oip } \xi)) = \text{val})))))" \end{aligned}$$

```

by(inv_cterms inv add: onl_invariant_sterms [OF aadv_wf rreq_oip])
show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aadv_wf rreq_oip]
  onl_invariant_sterms [OF aadv_wf rreq_dip_in_vD_dip_eq_ip]
  onl_invariant_sterms [OF aadv_wf rrep_prrep]
  onl_invariant_sterms [OF aadv_wf rreq_oip_kD]
  onl_invariant_sterms [OF aadv_wf rreq_dip_kD_oip_sqn])

```

qed

Proposition 7.15

lemma rerr\_invalid\_any\_step\_invariant:

"paadv i  $\models_A$  onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), a, \_)$ . anycast (rerr\_invalid (rt  $\xi$ )) a)"

proof -

```

have dests_inv: "paadv i  $\models$ 
  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ . ( $l \in \{PAadv-:15, PPkt-:7, PRreq-:11,$ 
    PRreq-:20, PRrep-:7, PRerr-:1}
     $\longrightarrow (\forall ip \in \text{dom}(\text{dests } \xi)$ .  $ip \in vD(rt \ \xi)$ ))
   $\wedge (l \in \{PAadv-:16..PAadv-:17\}$ 
     $\cup \{PPkt-:8..PPkt-:9\}$ 
     $\cup \{PRreq-:12..PRreq-:13\}$ 
     $\cup \{PRreq-:21..PRreq-:22\}$ 
     $\cup \{PRrep-:8..PRrep-:9\}$ 
     $\cup \{PRerr-:2..PRerr-:4\} \longrightarrow (\forall ip \in \text{dom}(\text{dests } \xi)$ .  $ip \in iD(rt \ \xi)$ 
       $\wedge \text{the } (\text{dests } \xi \text{ } ip) = \text{sqn } (rt \ \xi) \text{ } ip)$ ))
   $\wedge (l = PPkt-:12 \longrightarrow \text{dip } \xi \in iD(rt \ \xi))$ )"
```

by inv\_cterms (clarsimp split: split\_if\_asm option.split\_asm simp: domIff)+

show ?thesis

by (inv\_cterms inv add: onl\_invariant\_sterms [OF aadv\_wf dests\_inv])

qed

Proposition 7.16

Some well-definedness obligations are irrelevant for the Isabelle development:

1. In each routing table there is at most one entry for each destination: guaranteed by type.
2. In each store of queued data packets there is at most one data queue for each destination: guaranteed by structure.
3. Whenever a set of pairs ( $rip$ ,  $rsn$ ) is assigned to the variable  $dests$  of type  $ip \rightarrow sqn$ , or to the first argument of the function  $rerr$ , this set is a partial function, i.e., there is at most one entry ( $rip$ ,  $rsn$ ) for each destination  $rip$ : guaranteed by type.

lemma dests\_vD\_inc\_sqn:

```

"paadv i  $\models$ 
  onl  $\Gamma_{AODV}$  ( $\lambda(\xi, l)$ . ( $l \in \{PAadv-:15, PPkt-:7, PRreq-:11, PRreq-:20, PRrep-:7\}$ 
     $\longrightarrow (\forall ip \in \text{dom}(\text{dests } \xi)$ .  $ip \in vD(rt \ \xi) \wedge \text{the } (\text{dests } \xi \text{ } ip) = \text{inc } (\text{sqn } (rt \ \xi) \text{ } ip)$ ))
   $\wedge (l = PRerr-:1$ 
     $\longrightarrow (\forall ip \in \text{dom}(\text{dests } \xi)$ .  $ip \in vD(rt \ \xi) \wedge \text{the } (\text{dests } \xi \text{ } ip) > \text{sqn } (rt \ \xi) \text{ } ip)$ ))"
```

by inv\_cterms (clarsimp split: split\_if\_asm option.split\_asm)+

Proposition 7.27

lemma route\_tables\_fresher:

"paadv i  $\models_A$  (recvmgs rreq\_rrep\_sn  $\rightarrow$ ) onll  $\Gamma_{AODV}$  ( $\lambda((\xi, \_), \_, (\xi', \_))$ .

$\forall \text{dip} \in kD(rt \ \xi)$ .  $rt \ \xi \sqsubseteq_{\text{dip}} rt \ \xi'$ )"

proof (inv\_cterms inv add:

```

  onl_invariant_sterms [OF aadv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aadv_wf hop_count_positive [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aadv_wf osn_rreq]
  onl_invariant_sterms [OF aadv_wf dsn_rrep]

```

```

    onl_invariant_sterms [OF aadv_wf invariant_restrict_inD])
  fix  $\xi$  pp p'
  assume "( $\xi$ , pp)  $\in$  reachable (paadv i) (recvmmsg rreq_rrep_sn)"
    and "{PRreq-:2}[ $\lambda\xi$ .  $\xi$ (|rt := update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )))]
      p'  $\in$  sterms  $\Gamma_{AODV}$  pp"
    and "Suc 0  $\leq$  osn  $\xi$ "
    and *: " $\forall ip \in kD$  (rt  $\xi$ ). Suc 0  $\leq$  the (dhops (rt  $\xi$ ) ip)"
  show " $\forall ip \in kD$  (rt  $\xi$ ). rt  $\xi$   $\sqsubseteq_{ip}$  update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )"
  proof
    fix ip
    assume "ip  $\in kD$  (rt  $\xi$ )"
    moreover with * have "1  $\leq$  the (dhops (rt  $\xi$ ) ip)" by simp
    moreover from (Suc 0  $\leq$  osn  $\xi$ )
      have "update_arg_wf (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )" ..
    ultimately show "rt  $\xi$   $\sqsubseteq_{ip}$  update (rt  $\xi$ ) (oip  $\xi$ ) (osn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )"
      by (rule rt_fresher_update)
  qed
next
  fix  $\xi$  pp p'
  assume "( $\xi$ , pp)  $\in$  reachable (paadv i) (recvmmsg rreq_rrep_sn)"
    and "{PRrep-:0}[ $\lambda\xi$ .  $\xi$ (|rt := update (rt  $\xi$ ) (dip  $\xi$ ) (dsn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )))]
      p'  $\in$  sterms  $\Gamma_{AODV}$  pp"
    and "Suc 0  $\leq$  dsn  $\xi$ "
    and *: " $\forall ip \in kD$  (rt  $\xi$ ). Suc 0  $\leq$  the (dhops (rt  $\xi$ ) ip)"
  show " $\forall ip \in kD$  (rt  $\xi$ ). rt  $\xi$   $\sqsubseteq_{ip}$  update (rt  $\xi$ ) (dip  $\xi$ ) (dsn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )"
  proof
    fix ip
    assume "ip  $\in kD$  (rt  $\xi$ )"
    moreover with * have "1  $\leq$  the (dhops (rt  $\xi$ ) ip)" by simp
    moreover from (Suc 0  $\leq$  dsn  $\xi$ )
      have "update_arg_wf (dsn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )" ..
    ultimately show "rt  $\xi$   $\sqsubseteq_{ip}$  update (rt  $\xi$ ) (dip  $\xi$ ) (dsn  $\xi$ , kno, val, Suc (hops  $\xi$ ), sip  $\xi$ )"
      by (rule rt_fresher_update)
  qed
qed
end

```

## 5.7 The quality increases predicate

```

theory E_Quality_Increases
imports E_Aadv_Predicates E_Fresher
begin

```

```

definition quality_increases :: "state  $\Rightarrow$  state  $\Rightarrow$  bool"
where "quality_increases  $\xi$   $\xi'$   $\equiv$  ( $\forall dip \in kD$ (rt  $\xi$ ). dip  $\in kD$ (rt  $\xi'$ )  $\wedge$  rt  $\xi$   $\sqsubseteq_{dip}$  rt  $\xi'$ )
       $\wedge$  ( $\forall dip$ . sqn (rt  $\xi$ ) dip  $\leq$  sqn (rt  $\xi'$ ) dip)"

```

```

lemma quality_increasesI [intro!]:
  assumes " $\wedge dip$ . dip  $\in kD$ (rt  $\xi$ )  $\implies$  dip  $\in kD$ (rt  $\xi'$ )"
    and " $\wedge dip$ . [ $dip \in kD$ (rt  $\xi$ ); dip  $\in kD$ (rt  $\xi'$ ) ]  $\implies$  rt  $\xi$   $\sqsubseteq_{dip}$  rt  $\xi'$ "
    and " $\wedge dip$ . sqn (rt  $\xi$ ) dip  $\leq$  sqn (rt  $\xi'$ ) dip"
  shows "quality_increases  $\xi$   $\xi'$ "
  unfolding quality_increases_def using assms by clarsimp

```

```

lemma quality_increasesE [elim]:
  fixes dip
  assumes "quality_increases  $\xi$   $\xi'$ "
    and "dip  $\in kD$ (rt  $\xi$ )"
    and "[ $dip \in kD$ (rt  $\xi'$ ); rt  $\xi$   $\sqsubseteq_{dip}$  rt  $\xi'$ ; sqn (rt  $\xi$ ) dip  $\leq$  sqn (rt  $\xi'$ ) dip ]  $\implies$  R dip  $\xi$   $\xi'$ "
  shows "R dip  $\xi$   $\xi'$ "
  using assms unfolding quality_increases_def by clarsimp

```

```

lemma quality_increases_rt_fresherD [dest]:

```

```

fixes ip
assumes "quality_increases  $\xi \xi'$ "
  and "ip $\in$ kD(rt  $\xi$ )"
shows "rt  $\xi \sqsubseteq_{ip}$  rt  $\xi'$ "
using assms by auto

lemma quality_increases_sqnE [elim]:
  fixes dip
  assumes "quality_increases  $\xi \xi'$ "
    and "sqn (rt  $\xi$ ) dip  $\leq$  sqn (rt  $\xi'$ ) dip  $\implies$  R dip  $\xi \xi'$ "
  shows "R dip  $\xi \xi'$ "
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_refl [intro, simp]: "quality_increases  $\xi \xi$ "
  by rule simp_all

lemma strictly_fresher_quality_increases_right [elim]:
  fixes  $\sigma \sigma'$  dip
  assumes "rt ( $\sigma$  i)  $\sqsubseteq_{dip}$  rt ( $\sigma$  nhip)"
    and qinc: "quality_increases ( $\sigma$  nhip) ( $\sigma'$  nhip)"
    and "dip $\in$ kD(rt ( $\sigma$  nhip))"
  shows "rt ( $\sigma$  i)  $\sqsubseteq_{dip}$  rt ( $\sigma'$  nhip)"
proof -
  from qinc have "rt ( $\sigma$  nhip)  $\sqsubseteq_{dip}$  rt ( $\sigma'$  nhip)" using (dip $\in$ kD(rt ( $\sigma$  nhip)))
    by auto
  with (rt ( $\sigma$  i)  $\sqsubseteq_{dip}$  rt ( $\sigma$  nhip)) show ?thesis ..
qed

lemma kD_quality_increases [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "quality_increases  $\xi \xi'$ "
  shows "i $\in$ kD(rt  $\xi')$ "
using assms by auto

lemma kD_nsqn_quality_increases [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "quality_increases  $\xi \xi'$ "
  shows "i $\in$ kD(rt  $\xi')$   $\wedge$  nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi')$  i"
proof -
  from assms have "i $\in$ kD(rt  $\xi')$ " ..
  moreover with assms have "rt  $\xi \sqsubseteq_i$  rt  $\xi'$ " by auto
  ultimately have "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi')$  i"
    using (i $\in$ kD(rt  $\xi$ )) by - (erule(2) rt_fresher_imp_nsqn_le)
  with (i $\in$ kD(rt  $\xi')$ ) show ?thesis ..
qed

lemma nsqn_quality_increases [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "quality_increases  $\xi \xi'$ "
  shows "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi')$  i"
using assms by (rule kD_nsqn_quality_increases [THEN conjunct2])

lemma kD_nsqn_quality_increases_trans [elim]:
  assumes "i $\in$ kD(rt  $\xi$ )"
    and "s  $\leq$  nsqn (rt  $\xi$ ) i"
    and "quality_increases  $\xi \xi'$ "
  shows "i $\in$ kD(rt  $\xi')$   $\wedge$  s  $\leq$  nsqn (rt  $\xi')$  i"
proof
  from (i $\in$ kD(rt  $\xi$ )) and (quality_increases  $\xi \xi'$ ) show "i $\in$ kD(rt  $\xi')$ " ..
next
  from (i $\in$ kD(rt  $\xi$ )) and (quality_increases  $\xi \xi'$ ) have "nsqn (rt  $\xi$ ) i  $\leq$  nsqn (rt  $\xi')$  i" ..
  with (s  $\leq$  nsqn (rt  $\xi$ ) i) show "s  $\leq$  nsqn (rt  $\xi')$  i" by (rule le_trans)
qed

```



```

lemma nsqn_quality_increases_nsqn_lt_lt [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "s < nsqn (rt ξ) i"
  shows "s < nsqn (rt ξ') i"
proof -
  from assms(1-2) have "nsqn (rt ξ) i ≤ nsqn (rt ξ') i" ..
  with (s < nsqn (rt ξ) i) show "s < nsqn (rt ξ') i" by simp
qed

lemma nsqn_quality_increases_dhops [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "nsqn (rt ξ) i = nsqn (rt ξ') i"
  shows "the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i)"
using assms unfolding quality_increases_def
by (clarsimp) (drule(1) bspec, clarsimp simp: rt_fresher_def2)

lemma nsqn_quality_increases_nsqn_eq_le [elim]:
  assumes "i ∈ kD(rt ξ)"
    and "quality_increases ξ ξ'"
    and "s = nsqn (rt ξ) i"
  shows "s < nsqn (rt ξ') i ∨ (s = nsqn (rt ξ') i ∧ the (dhops (rt ξ) i) ≥ the (dhops (rt ξ') i))"
using assms by (metis nat_less_le nsqn_quality_increases nsqn_quality_increases_dhops)

lemma quality_increases_rreq_rrep_props [elim]:
  fixes sn ip hops sip
  assumes qinc: "quality_increases (σ sip) (σ' sip)"
    and "1 ≤ sn"
    and *: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip
      ∧ (nsqn (rt (σ sip)) ip = sn
        → (the (dhops (rt (σ sip)) ip) ≤ hops
          ∨ the (flag (rt (σ sip)) ip) = inv))"
  shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip
    ∧ (nsqn (rt (σ' sip)) ip = sn
      → (the (dhops (rt (σ' sip)) ip) ≤ hops
        ∨ the (flag (rt (σ' sip)) ip) = inv))"
  (is "_ ∧ ?nsqnafter")
proof -
  from * obtain "ip ∈ kD(rt (σ sip))" and "sn ≤ nsqn (rt (σ sip)) ip" by auto

  from ⟨quality_increases (σ sip) (σ' sip)⟩
  have "sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip" ..
  from ⟨quality_increases (σ sip) (σ' sip)⟩ and ⟨ip ∈ kD (rt (σ sip))⟩
  have "ip ∈ kD (rt (σ' sip))" ..

  from ⟨sn ≤ nsqn (rt (σ sip)) ip⟩ have ?nsqnafter
proof
  assume "sn < nsqn (rt (σ sip)) ip"
  also from ⟨ip ∈ kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  have "... ≤ nsqn (rt (σ' sip)) ip" ..
  finally have "sn < nsqn (rt (σ' sip)) ip" .
  thus ?thesis by simp
next
  assume "sn = nsqn (rt (σ sip)) ip"
  with ⟨ip ∈ kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  have "sn < nsqn (rt (σ' sip)) ip
    ∨ (sn = nsqn (rt (σ' sip)) ip
      ∧ the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip))" ..
  hence "sn < nsqn (rt (σ' sip)) ip
    ∨ (nsqn (rt (σ' sip)) ip = sn ∧ (the (dhops (rt (σ' sip)) ip) ≤ hops
      ∨ the (flag (rt (σ' sip)) ip) = inv))"
proof
  assume "sn < nsqn (rt (σ' sip)) ip" thus ?thesis ..

```

next

assume "sn = nsqn (rt (σ' sip)) ip  
    ∧ the (dhops (rt (σ sip)) ip) ≥ the (dhops (rt (σ' sip)) ip)"  
hence "sn = nsqn (rt (σ' sip)) ip"  
    and "the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)" by auto

from \* and ⟨sn = nsqn (rt (σ sip)) ip⟩ have "the (dhops (rt (σ sip)) ip) ≤ hops  
    ∨ the (flag (rt (σ sip)) ip) = inv"

by simp

thus ?thesis

proof

assume "the (dhops (rt (σ sip)) ip) ≤ hops"  
with ⟨the (dhops (rt (σ' sip)) ip) ≤ the (dhops (rt (σ sip)) ip)⟩  
    have "the (dhops (rt (σ' sip)) ip) ≤ hops" by simp  
with ⟨sn = nsqn (rt (σ' sip)) ip⟩ show ?thesis by simp

next

assume "the (flag (rt (σ sip)) ip) = inv"  
with ⟨ip ∈ kD(rt (σ sip))⟩ have "nsqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip - 1" ..

with ⟨sn ≥ 1⟩ and ⟨sn = nsqn (rt (σ sip)) ip⟩  
    have "sqn (rt (σ sip)) ip > 1" by simp

from ⟨ip ∈ kD(rt (σ' sip))⟩ show ?thesis

proof (rule vD\_or\_iD)

    assume "ip ∈ iD(rt (σ' sip))"  
    hence "the (flag (rt (σ' sip)) ip) = inv" ..  
    with ⟨sn = nsqn (rt (σ' sip)) ip⟩ show ?thesis  
        by simp

next

    assume "ip ∈ vD(rt (σ' sip))"  
    hence "nsqn (rt (σ' sip)) ip = sqn (rt (σ' sip)) ip" ..  
    with ⟨sqn (rt (σ sip)) ip ≤ sqn (rt (σ' sip)) ip⟩  
        have "nsqn (rt (σ' sip)) ip ≥ sqn (rt (σ sip)) ip" by simp

with ⟨sqn (rt (σ sip)) ip > 1⟩  
    have "nsqn (rt (σ' sip)) ip > sqn (rt (σ sip)) ip - 1" by simp  
with ⟨nsqn (rt (σ sip)) ip = sqn (rt (σ sip)) ip - 1⟩  
    have "nsqn (rt (σ' sip)) ip > nsqn (rt (σ sip)) ip" by simp  
with ⟨sn = nsqn (rt (σ sip)) ip⟩ have "nsqn (rt (σ' sip)) ip > sn"  
    by simp  
thus ?thesis ..

qed

qed

qed

thus ?thesis by (metis (mono\_tags) le\_cases not\_le)

qed

with ⟨ip ∈ kD (rt (σ' sip))⟩ show "ip ∈ kD (rt (σ' sip)) ∧ ?nsqnafter" ..

qed

lemma quality\_increases\_rreq\_rrep\_props':

fixes sn ip hops sip

assumes "∀j. quality\_increases (σ j) (σ' j)"

and "1 ≤ sn"

and \*: "ip ∈ kD(rt (σ sip)) ∧ sn ≤ nsqn (rt (σ sip)) ip  
    ∧ (nsqn (rt (σ sip)) ip = sn  
    → (the (dhops (rt (σ sip)) ip) ≤ hops  
        ∨ the (flag (rt (σ sip)) ip) = inv))"

shows "ip ∈ kD(rt (σ' sip)) ∧ sn ≤ nsqn (rt (σ' sip)) ip  
    ∧ (nsqn (rt (σ' sip)) ip = sn  
    → (the (dhops (rt (σ' sip)) ip) ≤ hops  
        ∨ the (flag (rt (σ' sip)) ip) = inv))"

proof -

from assms(1) have "quality\_increases (σ sip) (σ' sip)" ..

thus ?thesis using *assms(2-3)* by (rule *quality\_increases\_rreq\_rrep\_props*)  
qed

lemma *rteq\_quality\_increases*:

assumes " $\forall j. j \neq i \longrightarrow \text{quality\_increases } (\sigma j) (\sigma' j)$ "  
and " $\text{rt } (\sigma' i) = \text{rt } (\sigma i)$ "  
shows " $\forall j. \text{quality\_increases } (\sigma j) (\sigma' j)$ "  
using *assms* by *clarsimp* (*metis order\_refl quality\_increasesI rt\_fresher\_refl*)

definition *msg\_fresh* :: "(ip  $\Rightarrow$  state)  $\Rightarrow$  msg  $\Rightarrow$  bool"

where "*msg\_fresh*  $\sigma m \equiv$

case *m* of Rreq *hopsc* \_ \_ \_ *oipc* *osnc* *sipc* \_  $\Rightarrow$   $\text{osnc} \geq 1 \wedge (\text{sipc} \neq \text{oipc} \longrightarrow$   
 $\text{oipc} \in \text{kD}(\text{rt } (\sigma \text{sipc})) \wedge \text{nsqn } (\text{rt } (\sigma \text{sipc})) \text{oipc} \geq \text{osnc}$   
 $\wedge (\text{nsqn } (\text{rt } (\sigma \text{sipc})) \text{oipc} = \text{osnc}$   
 $\longrightarrow (\text{hopsc} \geq \text{the } (\text{dhops } (\text{rt } (\sigma \text{sipc})) \text{oipc})$   
 $\vee \text{the } (\text{flag } (\text{rt } (\sigma \text{sipc})) \text{oipc}) = \text{inv}))$   
| Rrep *hopsc* *dipc* *dsnc* \_ *sipc*  $\Rightarrow$   $\text{dsnc} \geq 1 \wedge (\text{sipc} \neq \text{dipc} \longrightarrow$   
 $\text{dipc} \in \text{kD}(\text{rt } (\sigma \text{sipc})) \wedge \text{nsqn } (\text{rt } (\sigma \text{sipc})) \text{dipc} \geq \text{dsnc}$   
 $\wedge (\text{nsqn } (\text{rt } (\sigma \text{sipc})) \text{dipc} = \text{dsnc}$   
 $\longrightarrow (\text{hopsc} \geq \text{the } (\text{dhops } (\text{rt } (\sigma \text{sipc})) \text{dipc})$   
 $\vee \text{the } (\text{flag } (\text{rt } (\sigma \text{sipc})) \text{dipc}) = \text{inv}))$   
| Rerr *destsc* *sipc*  $\Rightarrow$  ( $\forall \text{ripc} \in \text{dom}(\text{destsc}). (\text{ripc} \in \text{kD}(\text{rt } (\sigma \text{sipc}))$   
 $\wedge \text{the } (\text{destsc } \text{ripc}) - 1 \leq \text{nsqn } (\text{rt } (\sigma \text{sipc})) \text{ripc})$ )  
| \_  $\Rightarrow$  True"

lemma *msg\_fresh [simp]*:

" $\wedge \text{hops } \text{dip } \text{dsn } \text{dsk } \text{oip } \text{osn } \text{sip handled.}$   
*msg\_fresh*  $\sigma$  (Rreq *hops* *dip* *dsn* *dsk* *oip* *osn* *sip* handled) =  
 $(\text{osn} \geq 1 \wedge (\text{sip} \neq \text{oip} \longrightarrow \text{oip} \in \text{kD}(\text{rt } (\sigma \text{sip})))$   
 $\wedge \text{nsqn } (\text{rt } (\sigma \text{sip})) \text{oip} \geq \text{osn}$   
 $\wedge (\text{nsqn } (\text{rt } (\sigma \text{sip})) \text{oip} = \text{osn}$   
 $\longrightarrow (\text{hops} \geq \text{the } (\text{dhops } (\text{rt } (\sigma \text{sip})) \text{oip})$   
 $\vee \text{the } (\text{flag } (\text{rt } (\sigma \text{sip})) \text{oip}) = \text{inv}))$ "  
" $\wedge \text{hops } \text{dip } \text{dsn } \text{oip } \text{sip. } \text{msg\_fresh } \sigma$  (Rrep *hops* *dip* *dsn* *oip* *sip*) =  
 $(\text{dsn} \geq 1 \wedge (\text{sip} \neq \text{dip} \longrightarrow \text{dip} \in \text{kD}(\text{rt } (\sigma \text{sip})))$   
 $\wedge \text{nsqn } (\text{rt } (\sigma \text{sip})) \text{dip} \geq \text{dsn}$   
 $\wedge (\text{nsqn } (\text{rt } (\sigma \text{sip})) \text{dip} = \text{dsn}$   
 $\longrightarrow (\text{hops} \geq \text{the } (\text{dhops } (\text{rt } (\sigma \text{sip})) \text{dip})$   
 $\vee \text{the } (\text{flag } (\text{rt } (\sigma \text{sip})) \text{dip}) = \text{inv}))$ "  
" $\wedge \text{dests } \text{sip. } \text{msg\_fresh } \sigma$  (Rerr *dests* *sip*) =  
 $(\forall \text{ripc} \in \text{dom}(\text{dests}). (\text{ripc} \in \text{kD}(\text{rt } (\sigma \text{sip}))$   
 $\wedge \text{the } (\text{dests } \text{ripc}) - 1 \leq \text{nsqn } (\text{rt } (\sigma \text{sip})) \text{ripc}))$ "  
" $\wedge \text{d } \text{dip. } \text{msg\_fresh } \sigma$  (Newpkt *d* *dip*) = True"  
" $\wedge \text{d } \text{dip } \text{sip. } \text{msg\_fresh } \sigma$  (Pkt *d* *dip* *sip*) = True"  
unfolding *msg\_fresh\_def* by *simp\_all*

lemma *msg\_fresh\_inc\_sn [simp, elim]*:

"*msg\_fresh*  $\sigma m \Longrightarrow \text{rreq\_rrep\_sn } m$ "  
by (cases *m*) *simp\_all*

lemma *recv\_msg\_fresh\_inc\_sn [simp, elim]*:

"*orecvmsg* (*msg\_fresh*)  $\sigma m \Longrightarrow \text{recvmsg rreq\_rrep\_sn } m$ "  
by (cases *m*) *simp\_all*

lemma *rreq\_nsqn\_is\_fresh [simp]*:

fixes  $\sigma$  *msg* *hops* *dip* *dsn* *dsk* *oip* *osn* *sip* handled  
assumes "*rreq\_rrep\_fresh* (rt ( $\sigma$  *sip*)) (Rreq *hops* *dip* *dsn* *dsk* *oip* *osn* *sip* handled)"  
and "*rreq\_rrep\_sn* (Rreq *hops* *dip* *dsn* *dsk* *oip* *osn* *sip* handled)"  
shows "*msg\_fresh*  $\sigma$  (Rreq *hops* *dip* *dsn* *dsk* *oip* *osn* *sip* handled)"  
(is "*msg\_fresh*  $\sigma$  ?*msg*")

proof -

let ?*rt* = "rt ( $\sigma$  *sip*)"  
from *assms(2)* have " $1 \leq \text{osn}$ " by *simp*  
thus ?*thesis*

```

unfolding msg_fresh_def
proof (simp only: msg.case, intro conjI impI)
  assume "sip ≠ oip"
  with assms(1) show "oip ∈ kD(?rt)" by simp
next
assume "sip ≠ oip"
  and "nsqn ?rt oip = osn"
show "the (dhops ?rt oip) ≤ hops ∨ the (flag ?rt oip) = inv"
proof (cases "oip ∈ vD(?rt)")
  assume "oip ∈ vD(?rt)"
  hence "nsqn ?rt oip = sqn ?rt oip" ..
  with ⟨nsqn ?rt oip = osn⟩ have "sqn ?rt oip = osn" by simp
  with assms(1) and ⟨sip ≠ oip⟩ have "the (dhops ?rt oip) ≤ hops"
  by simp
  thus ?thesis ..
next
assume "oip ∉ vD(?rt)"
moreover from assms(1) and ⟨sip ≠ oip⟩ have "oip ∈ kD(?rt)" by simp
ultimately have "oip ∈ iD(?rt)" by auto
hence "the (flag ?rt oip) = inv" ..
thus ?thesis ..
qed
next
assume "sip ≠ oip"
with assms(1) have "osn ≤ sqn ?rt oip" by auto
thus "osn ≤ nsqn (rt (σ sip)) oip"
proof (rule nat_le_eq_or_lt)
  assume "osn < sqn ?rt oip"
  hence "osn ≤ sqn ?rt oip - 1" by simp
  also have "... ≤ nsqn ?rt oip" by (rule sqn_nsqn)
  finally show "osn ≤ nsqn ?rt oip" .
next
assume "osn = sqn ?rt oip"
with assms(1) and ⟨sip ≠ oip⟩ have "oip ∈ kD(?rt)"
  and "the (flag ?rt oip) = val"
  by auto
hence "nsqn ?rt oip = sqn ?rt oip" ..
with ⟨osn = sqn ?rt oip⟩ have "nsqn ?rt oip = osn" by simp
thus "osn ≤ nsqn ?rt oip" by simp
qed
qed simp
qed

lemma rrep_nsqn_is_fresh [simp]:
  fixes σ msg hops dip dsn oip sip
  assumes "rreq_rrep_fresh (rt (σ sip)) (Rrep hops dip dsn oip sip)"
  and "rreq_rrep_sn (Rrep hops dip dsn oip sip)"
  shows "msg_fresh σ (Rrep hops dip dsn oip sip)"
  (is "msg_fresh σ ?msg")
proof -
  let ?rt = "rt (σ sip)"
  from assms have "sip ≠ dip → dip ∈ kD(?rt) ∧ sqn ?rt dip = dsn ∧ the (flag ?rt dip) = val"
  by simp
  hence "sip ≠ dip → dip ∈ kD(?rt) ∧ nsqn ?rt dip ≥ dsn"
  by clarsimp
  with assms show "msg_fresh σ ?msg"
  by clarsimp
qed

lemma rerr_nsqn_is_fresh [simp]:
  fixes σ msg dests sip
  assumes "rerr_invalid (rt (σ sip)) (Rerr dests sip)"
  shows "msg_fresh σ (Rerr dests sip)"
  (is "msg_fresh σ ?msg")

```

```

proof -
  let ?rt = "rt (σ sip)"
  from assms have *: "(∀rip∈dom(dests). (rip∈iD(rt (σ sip))
    ∧ the (dests rip) = sqn (rt (σ sip)) rip))"
    by clarsimp
  have "(∀rip∈dom(dests). (rip∈kD(rt (σ sip))
    ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip))"
proof
  fix rip
  assume "rip ∈ dom dests"
  with * have "rip∈iD(rt (σ sip))" and "the (dests rip) = sqn (rt (σ sip)) rip"
    by auto

  from this(2) have "the (dests rip) - 1 = sqn (rt (σ sip)) rip - 1" by simp
  also have "... ≤ nsqn (rt (σ sip)) rip" by (rule sqn_nsqn)
  finally have "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip" .

  with ⟨rip∈iD(rt (σ sip))⟩
    show "rip∈kD(rt (σ sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
      by clarsimp
qed
thus "msg_fresh σ ?msg"
  by simp
qed

```

```

lemma quality_increases_msg_fresh [elim]:
  assumes qinc: "∀j. quality_increases (σ j) (σ' j)"
    and "msg_fresh σ m"
  shows "msg_fresh σ' m"
using assms(2)
proof (cases m)
  fix hops dip dsn dsk oip osn sip handled
  assume [simp]: "m = Rreq hops dip dsn dsk oip osn sip handled"
    and "msg_fresh σ m"
  then have "osn ≥ 1" and "sip = oip ∨ (oip∈kD(rt (σ sip)) ∧ osn ≤ nsqn (rt (σ sip)) oip
    ∧ (nsqn (rt (σ sip)) oip = osn
    → (the (dhops (rt (σ sip)) oip) ≤ hops
    ∨ the (flag (rt (σ sip)) oip) = inv)))"

    by auto
  from this(2) show ?thesis
proof
  assume "sip = oip" with ⟨osn ≥ 1⟩ show ?thesis by simp
next
  assume "oip∈kD(rt (σ sip)) ∧ osn ≤ nsqn (rt (σ sip)) oip
    ∧ (nsqn (rt (σ sip)) oip = osn
    → (the (dhops (rt (σ sip)) oip) ≤ hops
    ∨ the (flag (rt (σ sip)) oip) = inv))"
  moreover from qinc have "quality_increases (σ sip) (σ' sip)" ..
  ultimately have "oip∈kD(rt (σ' sip)) ∧ osn ≤ nsqn (rt (σ' sip)) oip
    ∧ (nsqn (rt (σ' sip)) oip = osn
    → (the (dhops (rt (σ' sip)) oip) ≤ hops
    ∨ the (flag (rt (σ' sip)) oip) = inv))"

    using ⟨osn ≥ 1⟩ by (rule quality_increases_rreq_rrep_props [rotated 2])
  with ⟨osn ≥ 1⟩ show "msg_fresh σ' m"
    by (clarsimp)
qed
next
  fix hops dip dsn oip sip
  assume [simp]: "m = Rrep hops dip dsn oip sip"
    and "msg_fresh σ m"
  then have "dsn ≥ 1" and "sip = dip ∨ (dip∈kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip
    ∧ (nsqn (rt (σ sip)) dip = dsn
    → (the (dhops (rt (σ sip)) dip) ≤ hops
    ∨ the (flag (rt (σ sip)) dip) = inv)))"

```

```

by auto
from this(2) show "?thesis"
proof
  assume "sip = dip" with ⟨dsn ≥ 1⟩ show ?thesis by simp
next
  assume "dip ∈ kD(rt (σ sip)) ∧ dsn ≤ nsqn (rt (σ sip)) dip
        ∧ (nsqn (rt (σ sip)) dip = dsn
           → (the (dhops (rt (σ sip)) dip) ≤ hops
              ∨ the (flag (rt (σ sip)) dip) = inv))"
  moreover from qinc have "quality_increases (σ sip) (σ' sip)" ..
  ultimately have "dip ∈ kD(rt (σ' sip)) ∧ dsn ≤ nsqn (rt (σ' sip)) dip
                 ∧ (nsqn (rt (σ' sip)) dip = dsn
                    → (the (dhops (rt (σ' sip)) dip) ≤ hops
                       ∨ the (flag (rt (σ' sip)) dip) = inv))"

  using ⟨dsn ≥ 1⟩ by (rule quality_increases_rreq_rrep_props [rotated 2])
  with ⟨dsn ≥ 1⟩ show "msg_fresh σ' m"
  by clarsimp
qed
next
  fix dests sip
  assume [simp]: "m = Rerr dests sip"
  and "msg_fresh σ m"
  then have *: "∀ rip ∈ dom(dests). rip ∈ kD(rt (σ sip))
              ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"

  by simp
  have "∀ rip ∈ dom(dests). rip ∈ kD(rt (σ' sip))
        ∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip"

  proof
    fix rip
    assume "rip ∈ dom(dests)"
    with * have "rip ∈ kD(rt (σ sip))" and "the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
      by - (drule(1) bspec, clarsimp)+
    moreover from qinc have "quality_increases (σ sip) (σ' sip)" by simp
    ultimately show "rip ∈ kD(rt (σ' sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ' sip)) rip" ..
  qed
  thus ?thesis by simp
qed simp_all

```

end

## 5.8 The ‘open’ AODV model

```

theory E_OAodv
imports E_Aodv "../.../AWN/OAWN_SOS_Labels" "../.../AWN/OAWN_Convert"
begin

```

Definitions for stating and proving global network properties over individual processes.

```

definition σAODV' :: "(ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp) set"
where "σAODV' ≡ {⟨λi. aodv_init i, ΓAODV PAodv⟩}"

```

**abbreviation** opaodv

```

:: "ip ⇒ ((ip ⇒ state) × (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
where
"opaodv i ≡ (| init = σAODV', trans = oseqp_sos ΓAODV i |)"

```

```

lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))"
unfolding σAODV'_def σAODV'_def by rule simp_all

```

```

lemma oaodv_control_within [simp]: "control_within ΓAODV (init (opaodv i))"
unfolding σAODV'_def by (rule control_withinI) (auto simp del: ΓAODV_simps)

```

```

lemma σAODV'_labels [simp]: "(σ, p) ∈ σAODV' ⇒ labels ΓAODV p = {PAodv-.0}"
unfolding σAODV'_def by simp

```

```

lemma oaodv_init_kD_empty [simp]:
  "( $\sigma$ , p)  $\in$   $\sigma_{AODV}' \implies kD$  (rt ( $\sigma$  i)) = {}"
  unfolding  $\sigma_{AODV}'$ _def kD_def by simp

lemma oaodv_init_vD_empty [simp]:
  "( $\sigma$ , p)  $\in$   $\sigma_{AODV}' \implies vD$  (rt ( $\sigma$  i)) = {}"
  unfolding  $\sigma_{AODV}'$ _def vD_def by simp

lemma oaodv_trans: "trans (opaodv i) = oseqp_sos  $\Gamma_{AODV}$  i"
  by simp

declare
  oseq_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]
  oseq_step_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels oaodv_trans, cterms_intros]

end

```

## 5.9 Global invariant proofs over sequential processes

```

theory E_Global_Invariants
imports E_Seq_Invariants
  E_Aodv_Predicates
  E_Fresher
  E_Quality_Increases
  "../.../AWN/DAWN_Convert"
  E_OAodv
begin

lemma other_quality_increases [elim]:
  assumes "other quality_increases I  $\sigma$   $\sigma'$ "
  shows " $\forall j$ . quality_increases ( $\sigma$  j) ( $\sigma'$  j)"
  using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
  fixes m
  assumes *: "otherwith P I (orecvmsg Q)  $\sigma$   $\sigma'$  a"
  and weakenP: " $\bigwedge \sigma m$ . P  $\sigma$  m  $\implies$  P'  $\sigma$  m"
  and weakenQ: " $\bigwedge \sigma m$ . Q  $\sigma$  m  $\implies$  Q'  $\sigma$  m"
  shows "otherwith P' I (orecvmsg Q')  $\sigma$   $\sigma'$  a"
proof
  fix j
  assume "j  $\notin$  I"
  with * have "P ( $\sigma$  j) ( $\sigma'$  j)" by auto
  thus "P' ( $\sigma$  j) ( $\sigma'$  j)" by (rule weakenP)
next
  from * have "orecvmsg Q  $\sigma$  a" by auto
  thus "orecvmsg Q'  $\sigma$  a"
  by rule (erule weakenQ)
qed

lemma oreceived_msg_inv:
  assumes other: " $\bigwedge \sigma \sigma' m$ . [ $P$   $\sigma$  m; other Q {i}  $\sigma$   $\sigma'$ ]  $\implies$  P  $\sigma'$  m"
  and local: " $\bigwedge \sigma m$ . P  $\sigma$  m  $\implies$  P ( $\sigma$ (i :=  $\sigma$  i(msg := m))) m"
  shows "opaodv i  $\models$  (otherwith Q {i} (orecvmsg P), other Q {i}  $\rightarrow$ )
    onl  $\Gamma_{AODV}$  ( $\lambda(\sigma, l)$ . l  $\in$  {PAodv-:1}  $\longrightarrow$  P  $\sigma$  (msg ( $\sigma$  i)))"
proof (inv_cterms, intro impI)
  fix  $\sigma$   $\sigma'$  l
  assume "l = PAodv-:1  $\longrightarrow$  P  $\sigma$  (msg ( $\sigma$  i))"
  and "l = PAodv-:1"
  and "other Q {i}  $\sigma$   $\sigma'$ "
  from this(1-2) have "P  $\sigma$  (msg ( $\sigma$  i))" ..
  hence "P  $\sigma'$  (msg ( $\sigma$  i))" using (other Q {i}  $\sigma$   $\sigma'$ )
  by (rule other)
  moreover from (other Q {i}  $\sigma$   $\sigma'$ ) have " $\sigma'$  i =  $\sigma$  i" ..

```

```

ultimately show "P σ' (msg (σ' i))" by simp
next
fix σ σ' msg
assume "otherwith Q {i} (orecvmsg P) σ σ' (receive msg)"
  and "σ' i = σ i(msg := msg)"
from this(1) have "P σ msg"
  and "∀j. j≠i → Q (σ j) (σ' j)" by auto
from this(1) have "P (σ(i := σ i(msg := msg))) msg" by (rule local)
thus "P σ' msg"
proof (rule other)
  from ⟨σ' i = σ i(msg := msg)⟩ and ⟨∀j. j≠i → Q (σ j) (σ' j)⟩
  show "other Q {i} (σ(i := σ i(msg := msg))) σ'"
  by - (rule otherI, auto)
qed
qed

```

(Equivalent to) Proposition 7.27

lemma local\_quality\_increases:

```

"paadv i ⊨A (recvmsg rreq_rrep_sn →) onll ΓAODV (λ((ξ, _), _, (ξ', _)). quality_increases ξ ξ')"
proof (rule step_invariantI)
  fix s a s'
  assume sr: "s ∈ reachable (paadv i) (recvmsg rreq_rrep_sn)"
  and tr: "(s, a, s') ∈ trans (paadv i)"
  and rm: "recvmsg rreq_rrep_sn a"
  from sr have srTT: "s ∈ reachable (paadv i) TT" ..

  from route_tables_fresher sr tr rm
  have "onll ΓAODV (λ((ξ, _), _, (ξ', _)). ∀dip∈kD (rt ξ). rt ξ ⊆dip rt ξ') (s, a, s')"
  by (rule step_invariantD)

  moreover from known_destinations_increase srTT tr TT_True
  have "onll ΓAODV (λ((ξ, _), _, (ξ', _)). kD (rt ξ) ⊆ kD (rt ξ')) (s, a, s')"
  by (rule step_invariantD)

  moreover from sqns_increase srTT tr TT_True
  have "onll ΓAODV (λ((ξ, _), _, (ξ', _)). ∀ip. sqn (rt ξ) ip ≤ sqn (rt ξ') ip) (s, a, s')"
  by (rule step_invariantD)

  ultimately show "onll ΓAODV (λ((ξ, _), _, (ξ', _)). quality_increases ξ ξ') (s, a, s')"
  unfolding onll_def by auto
qed

```

lemmas olocal\_quality\_increases =

```

open_seq_step_invariant [OF local_quality_increases initiali_aadv oaadv_trans aadv_trans,
  simplified seql_onll_swap]

```

lemma oquality\_increases:

```

"opaadv i ⊨A (otherwith quality_increases {i} (orecvmsg (λ_. rreq_rrep_sn)),
  other quality_increases {i} →)
  onll ΓAODV (λ((σ, _), _, (σ', _)). ∀j. quality_increases (σ j) (σ' j))"
(is "_ ⊨A (?S, _ →) _")
proof (rule onll_ostep_invariantI, simp)
  fix σ p l a σ' p' l'
  assume or: "(σ, p) ∈ oreachable (opaadv i) ?S (other quality_increases {i})"
  and ll: "l ∈ labels ΓAODV p"
  and "?S σ σ' a"
  and tr: "((σ, p), a, (σ', p')) ∈ oseqp_sos ΓAODV i"
  and ll': "l' ∈ labels ΓAODV p'"
  from this(1-3) have "orecvmsg (λ_. rreq_rrep_sn) σ a"
  by (auto dest!: oreachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"
    and QU="other quality_increases {i}"]
    otherwith_actionD)
  with or have orw: "(σ, p) ∈ oreachable (opaadv i) (act (recvmsg rreq_rrep_sn))
    (other quality_increases {i})"

```



```

by - (erule oreachable_weakenE, auto)
with tr l1 l1' and ⟨orecvmsg (λ_. rreq_rrep_sn) σ a⟩ have "quality_increases (σ i) (σ' i)"
by - (drule onll_ostep_invariantD [OF olocal_quality_increases], auto simp: seqll_def)
with ⟨?S σ σ' a⟩ show "∀j. quality_increases (σ j) (σ' j)"
by (auto dest!: otherwith_syncD)
qed

```

lemma rreq\_rrep\_nsqn\_fresh\_any\_step\_invariant:

```

"opaadv i ⊨A (act (recvmsg rreq_rrep_sn), other A {i} →)
  onll ΓAODV (λ((σ, _), a, _). anycast (msg_fresh σ) a)"
proof (rule ostep_invariantI, simp del: act_simp)
  fix σ p a σ' p'
  assume or: "(σ, p) ∈ oreachable (opaadv i) (act (recvmsg rreq_rrep_sn)) (other A {i})"
  and "((σ, p), a, (σ', p')) ∈ oseqp_sos ΓAODV i"
  and recv: "act (recvmsg rreq_rrep_sn) σ σ' a"
  obtain l l' where "l ∈ labels ΓAODV p" and "l' ∈ labels ΓAODV p'"
  by (metis aadv_ex_label)
  from ⟨((σ, p), a, (σ', p')) ∈ oseqp_sos ΓAODV i⟩
  have tr: "((σ, p), a, (σ', p')) ∈ trans (opaadv i)" by simp

  have "anycast (rreq_rrep_fresh (rt (σ i))) a"
  proof -
    have "opaadv i ⊨A (act (recvmsg rreq_rrep_sn), other A {i} →)
      onll ΓAODV (seqll i (λ((ξ, _), a, _). anycast (rreq_rrep_fresh (rt ξ)) a))"
    by (rule ostep_invariant_weakenE [OF
      open_seq_step_invariant [OF rreq_rrep_fresh_any_step_invariant initiali_aadv,
        simplified seqll_onll_swap]]) auto
    hence "onll ΓAODV (seqll i (λ((ξ, _), a, _). anycast (rreq_rrep_fresh (rt ξ)) a))
      ((σ, p), a, (σ', p'))"
    using or tr recv by - (erule(4) ostep_invariantE)
    thus ?thesis
    using ⟨l ∈ labels ΓAODV p⟩ and ⟨l' ∈ labels ΓAODV p'⟩ by auto
  qed

```

moreover have "anycast (rerr\_invalid (rt (σ i))) a"

```

proof -
  have "opaadv i ⊨A (act (recvmsg rreq_rrep_sn), other A {i} →)
    onll ΓAODV (seqll i (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a))"
  by (rule ostep_invariant_weakenE [OF
    open_seq_step_invariant [OF rerr_invalid_any_step_invariant initiali_aadv,
      simplified seqll_onll_swap]]) auto
  hence "onll ΓAODV (seqll i (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a))
    ((σ, p), a, (σ', p'))"
  using or tr recv by - (erule(4) ostep_invariantE)
  thus ?thesis
  using ⟨l ∈ labels ΓAODV p⟩ and ⟨l' ∈ labels ΓAODV p'⟩ by auto
qed

```

moreover have "anycast rreq\_rrep\_sn a"

```

proof -
  from or tr recv
  have "onll ΓAODV (seqll i (λ(_, a, _). anycast rreq_rrep_sn a)) ((σ, p), a, (σ', p'))"
  by (rule ostep_invariantE [OF
    open_seq_step_invariant [OF rreq_rrep_sn_any_step_invariant initiali_aadv
      oaadv_trans aadv_trans,
      simplified seqll_onll_swap]])
  thus ?thesis
  using ⟨l ∈ labels ΓAODV p⟩ and ⟨l' ∈ labels ΓAODV p'⟩ by auto
qed

```

moreover have "anycast (λm. not\_Pkt m → msg\_sender m = i) a"

```

proof -
  have "opaadv i ⊨A (act (recvmsg rreq_rrep_sn), other A {i} →)
    onll ΓAODV (seqll i (λ((ξ, _), a, _). anycast (λm. not_Pkt m → msg_sender m = i) a))"

```

```

    by (rule ostep_invariant_weakenE [OF
      open_seq_step_invariant [OF sender_ip_valid initiali_aodv,
        simplified seqll_onll_swap]]) auto
  thus ?thesis using or tr recv ⟨l∈labels ΓAODV p⟩ and ⟨l'∈labels ΓAODV p'⟩
    by - (drule(3) onll_ostep_invariantD, auto)
qed

ultimately have "anycast (msg_fresh σ) a"
  by (simp_all add: anycast_def
      del: msg_fresh
      split: seq_action.split_asm msg.split_asm) simp_all
thus "onll ΓAODV (λ((σ, _), a, _). anycast (msg_fresh σ) a) ((σ, p), a, (σ', p'))"
  by auto
qed

lemma oreceived_rreq_rrep_nsqn_fresh_inv:
  "opaodv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onl ΓAODV (λ(σ, l). l ∈ {PAodv-:1} → msg_fresh σ (msg (σ i)))"
proof (rule oreceived_msg_inv)
  fix σ σ' m
  assume *: "msg_fresh σ m"
  and "other quality_increases {i} σ σ'"
  from this(2) have "∀j. quality_increases (σ j) (σ' j)" ..
  thus "msg_fresh σ' m" using * ..
next
  fix σ m
  assume "msg_fresh σ m"
  thus "msg_fresh (σ(i := σ i (msg := m))) m"
proof (cases m)
  fix dests sip
  assume "m = Rerr dests sip"
  with ⟨msg_fresh σ m⟩ show ?thesis by auto
qed auto
qed

lemma oquality_increases_nsqn_fresh:
  "opaodv i ⊨A (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onll ΓAODV (λ((σ, _), _, (σ', _)). ∀j. quality_increases (σ j) (σ' j))"
  by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
  "opaodv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onl ΓAODV (seq1 i (λ(ξ, l). l ∈ {PAodv-:4, PAodv-:5} ∪ {PRreq-:n | n. True} → 1 ≤ osn ξ))"
  by (rule oinvariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_aodv]])
  (auto simp: seq1_onll_swap)

lemma rreq_sip:
  "opaodv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onl ΓAODV (λ(σ, l).
      (l ∈ {PAodv-:4, PAodv-:5, PRreq-:0, PRreq-:2} ∧ sip (σ i) ≠ oip (σ i))
      → oip (σ i) ∈ kD(rt (σ (sip (σ i))))
      ∧ nsqn (rt (σ (sip (σ i)))) (oip (σ i)) ≥ osn (σ i)
      ∧ (nsqn (rt (σ (sip (σ i)))) (oip (σ i)) = osn (σ i)
      → (hops (σ i) ≥ the (dhops (rt (σ (sip (σ i)))) (oip (σ i)))
      ∨ the (flag (rt (σ (sip (σ i)))) (oip (σ i))) = inv)))"
  (is "_ ⊨ (?S, ?U →) _")
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh
  aodv_wf oaodv_trans]
  onl_oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
  onl_oinvariant_sterms [OF aodv_wf oosn_rreq])

```

```

      simp add: seqlsimp
      simp del: One_nat_def, rule impI)
fix  $\sigma \sigma' p l$ 
assume "( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U"
  and "l  $\in$  labels  $\Gamma_{AODV}$  p"
  and pre:
    "(l = PAadv-:4  $\vee$  l = PAadv-:5  $\vee$  l = PRreq-:0  $\vee$  l = PRreq-:2)  $\wedge$  sip ( $\sigma$  i)  $\neq$  oip ( $\sigma$  i)
     $\rightarrow$  oip ( $\sigma$  i)  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
     $\wedge$  osn ( $\sigma$  i)  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma$  i))
     $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma$  i)) = osn ( $\sigma$  i)
     $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma$  i)))  $\leq$  hops ( $\sigma$  i)
     $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma$  i))) = inv)"
  and "other quality_increases {i}  $\sigma \sigma'$ "
  and hyp: "(l=PAadv-:4  $\vee$  l=PAadv-:5  $\vee$  l=PRreq-:0  $\vee$  l=PRreq-:2)  $\wedge$  sip ( $\sigma'$  i)  $\neq$  oip ( $\sigma'$  i)"
  (is "?labels  $\wedge$  sip ( $\sigma'$  i)  $\neq$  oip ( $\sigma'$  i)")
from this(4) have " $\sigma' i = \sigma i$ " ..
with hyp have hyp': "?labels  $\wedge$  sip ( $\sigma$  i)  $\neq$  oip ( $\sigma$  i)" by simp
show "oip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma'$  (sip ( $\sigma' i$ ))))"
   $\wedge$  osn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (oip ( $\sigma' i$ ))
   $\wedge$  (nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (oip ( $\sigma' i$ )) = osn ( $\sigma' i$ )
   $\rightarrow$  the (dhops (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (oip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
   $\vee$  the (flag (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (oip ( $\sigma' i$ ))) = inv)"
proof (cases "sip ( $\sigma$  i) = i")
  assume "sip ( $\sigma$  i)  $\neq$  i"
  from (other quality_increases {i}  $\sigma \sigma'$ )
    have "quality_increases ( $\sigma$  (sip ( $\sigma$  i))) ( $\sigma'$  (sip ( $\sigma' i$ )))"
      by (rule otherE) (clarsimp simp: (sip ( $\sigma$  i)  $\neq$  i))
  moreover from (( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U) (l  $\in$  labels  $\Gamma_{AODV}$  p) and hyp
    have "l  $\leq$  osn ( $\sigma' i$ )"
      by (auto dest!: onl_oinvariant_weakenD [OF oosn_rreq]
          simp add: seqlsimp ( $\sigma' i = \sigma i$ ))
  moreover from (sip ( $\sigma$  i)  $\neq$  i) hyp' and pre
    have "oip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))"
       $\wedge$  osn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma' i$ ))
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma' i$ )) = osn ( $\sigma' i$ )
       $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
       $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (oip ( $\sigma' i$ ))) = inv)"
    by (auto simp: ( $\sigma' i = \sigma i$ ))
  ultimately show ?thesis
    by (rule quality_increases_rreq_rrep_props)
next
  assume "sip ( $\sigma$  i) = i" thus ?thesis
    using ( $\sigma' i = \sigma i$ ) hyp and pre by auto
qed
qed (auto elim!: quality_increases_rreq_rrep_props')

```

```

lemma odsn_rrep:
  "opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i}  $\rightarrow$ )
    onl  $\Gamma_{AODV}$  (seq1 i ( $\lambda(\xi, l). l \in \{PAadv-:6, PAadv-:7\} \cup \{PRrep-:n/n. True\} \rightarrow 1 \leq dsn \xi$ ))"
  by (rule oinvariant_weakenE [OF open_seq_invariant [OF dsn_rrep initiali_aadv]])
  (auto simp: seq1_onl_swap)

```

```

lemma rrep_sip:
  "opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i}  $\rightarrow$ )
    onl  $\Gamma_{AODV}$  ( $\lambda(\sigma, l).$ 
      (l  $\in$  {PAadv-:6, PAadv-:7, PRrep-:0, PRrep-:1}  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i))
       $\rightarrow$  dip ( $\sigma$  i)  $\in$  kD(rt ( $\sigma$  (sip ( $\sigma$  i))))
       $\wedge$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))  $\geq$  dsn ( $\sigma$  i)
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i)) = dsn ( $\sigma$  i)
       $\rightarrow$  (hops ( $\sigma$  i)  $\geq$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))))
       $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = inv))"
  (is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")

```

```

proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
                                                    oaadv_trans]
      onl_oInvariant_sterms [OF aadv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
      onl_oInvariant_sterms [OF aadv_wf odsn_rrep]
      simp del: One_nat_def, rule impI)
fix  $\sigma \sigma' p l$ 
assume "( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U"
  and "l  $\in$  labels  $\Gamma_{AODV} p$ "
  and pre:
    "(l = PAadv-:6  $\vee$  l = PAadv-:7  $\vee$  l = PRrep-:0  $\vee$  l = PRrep-:1)  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i)
     $\rightarrow$  dip ( $\sigma$  i)  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))
       $\wedge$  dsn ( $\sigma$  i)  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = dsn ( $\sigma$  i)
       $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i)))  $\leq$  hops ( $\sigma$  i)
         $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma$  i))) = inv"
  and "other quality_increases {i}  $\sigma \sigma'$ "
  and hyp: "(l=PAadv-:6  $\vee$  l=PAadv-:7  $\vee$  l=PRrep-:0  $\vee$  l=PRrep-:1)  $\wedge$  sip ( $\sigma'$  i)  $\neq$  dip ( $\sigma'$  i)"
    (is "?labels  $\wedge$  sip ( $\sigma'$  i)  $\neq$  dip ( $\sigma'$  i)")
from this(4) have " $\sigma' i = \sigma i$ " ..
with hyp have hyp': "?labels  $\wedge$  sip ( $\sigma$  i)  $\neq$  dip ( $\sigma$  i)" by simp
show "dip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma'$  (sip ( $\sigma' i$ ))))"
   $\wedge$  dsn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ ))
   $\wedge$  (nsqn (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ ))) = dsn ( $\sigma' i$ )
   $\rightarrow$  the (dhops (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
     $\vee$  the (flag (rt ( $\sigma'$  (sip ( $\sigma' i$ )))) (dip ( $\sigma' i$ ))) = inv"
proof (cases "sip ( $\sigma$  i) = i")
  assume "sip ( $\sigma$  i)  $\neq$  i"
  from (other quality_increases {i}  $\sigma \sigma'$ )
    have "quality_increases ( $\sigma$  (sip ( $\sigma$  i))) ( $\sigma'$  (sip ( $\sigma' i$ )))"
      by (rule otherE) (clarsimp simp: (sip ( $\sigma$  i)  $\neq$  i))
  moreover from (( $\sigma, p$ )  $\in$  oreachable (opaadv i) ?S ?U) (l  $\in$  labels  $\Gamma_{AODV} p$ ) and hyp
    have "l  $\leq$  dsn ( $\sigma' i$ )"
      by (auto dest!: onl_oInvariant_weakenD [OF odsn_rrep]
          simp add: seqsimp (sip ( $\sigma$  i) = i))
  moreover from (sip ( $\sigma$  i)  $\neq$  i) hyp' and pre
    have "dip ( $\sigma' i$ )  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))"
       $\wedge$  dsn ( $\sigma' i$ )  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ ))
       $\wedge$  (nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ ))) = dsn ( $\sigma' i$ )
       $\rightarrow$  the (dhops (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ )))  $\leq$  hops ( $\sigma' i$ )
         $\vee$  the (flag (rt ( $\sigma$  (sip ( $\sigma$  i)))) (dip ( $\sigma' i$ ))) = inv"
    by (auto simp: (sip ( $\sigma$  i) = i))
  ultimately show ?thesis
    by (rule quality_increases_rreq_rrep_props)
next
  assume "sip ( $\sigma$  i) = i" thus ?thesis
    using (sip ( $\sigma$  i) = i) hyp and pre by auto
qed
qed (auto simp add: seqsimp elim!: quality_increases_rreq_rrep_props')

```

lemma rerr\_sip:

```

"opaadv i  $\models$  (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i}  $\rightarrow$ )
  onl  $\Gamma_{AODV} (\lambda(\sigma, l).$ 
    l  $\in$  {PAadv-:8, PAadv-:9, PRerr-:0, PRerr-:1}
     $\rightarrow$  ( $\forall ripc \in dom(dests (\sigma i)). ripc \in kD(rt (\sigma (sip (\sigma i)))) \wedge$ 
      the (dests ( $\sigma$  i) ripc) - 1  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) ripc))"
(is "_  $\models$  (?S, ?U  $\rightarrow$ ) _")

```

proof -

```

{ fix dests rip sip rsn and  $\sigma \sigma' :: "ip \Rightarrow state"$ 
  assume qinc: " $\forall j. quality\_increases (\sigma j) (\sigma' j)"$ 
    and *: " $\forall rip \in dom dests. rip \in kD (rt (\sigma sip))$ 
       $\wedge$  the (dests rip) - 1  $\leq$  nsqn (rt ( $\sigma sip$ )) rip"
    and "dests rip = Some rsn"
  from this(3) have "rip  $\in$  dom dests" by auto

```

```

with * and ⟨dests rip = Some rsn⟩ have "rip∈kD(rt (σ sip))"
      and "rsn - 1 ≤ nsqn (rt (σ sip)) rip"

  by (auto dest!: bspec)
from qinc have "quality_increases (σ sip) (σ' sip)" ..
have "rip ∈ kD(rt (σ' sip)) ∧ rsn - 1 ≤ nsqn (rt (σ' sip)) rip"
proof
  from ⟨rip∈kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  show "rip ∈ kD(rt (σ' sip))" ..
next
  from ⟨rip∈kD(rt (σ sip))⟩ and ⟨quality_increases (σ sip) (σ' sip)⟩
  have "nsqn (rt (σ sip)) rip ≤ nsqn (rt (σ' sip)) rip" ..
  with ⟨rsn - 1 ≤ nsqn (rt (σ sip)) rip⟩ show "rsn - 1 ≤ nsqn (rt (σ' sip)) rip"
  by (rule le_trans)
qed
} note partial = this

show ?thesis
  by (inv_terms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
      oaadv_trans]
      onl_oinvariant_sterms [OF aadv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
      other_quality_increases other_localD
      simp del: One_nat_def, intro conjI]
      (clarsimp simp del: One_nat_def split: split_if_asm option.split_asm, erule(2) partial)+
qed

lemma prerr_guard: "paadv i ⊨
  onl ΓAODV (λ(ξ, l). (l = PRerr-:1
    → (∀ip∈dom(dests ξ). ip∈vD(rt ξ)
      ∧ the (nhop (rt ξ) ip) = sip ξ
      ∧ sqn (rt ξ) ip < the (dests ξ ip))))"
  by (inv_terms) (clarsimp split: option.split_asm split_if_asm)

lemmas odests_vD_inc_sqn =
  open_seq_invariant [OF dests_vD_inc_sqn initiali_aadv oaadv_trans aadv_trans,
    simplified seq_l_onl_swap,
    THEN oinvariant_anyact]

lemmas oprerr_guard =
  open_seq_invariant [OF prerr_guard initiali_aadv oaadv_trans aadv_trans,
    simplified seq_l_onl_swap,
    THEN oinvariant_anyact]

Proposition 7.28

lemma seq_compare_next_hop':
  "opaadv i ⊨ (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →) onl ΓAODV (λ(σ, _).
    ∀dip. let nhop = the (nhop (rt (σ i)) dip)
      in dip ∈ kD(rt (σ i)) ∧ nhop ≠ dip →
      dip ∈ kD(rt (σ nhop)) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhop)) dip)"
  (is "_ ⊨ (?S, ?U →) _")
proof -
{ fix nhop and σ σ' :: "ip ⇒ state"
  assume pre: "∀dip∈kD(rt (σ i)). nhop dip ≠ dip →
    dip∈kD(rt (σ (nhop dip))) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"
  and qinc: "∀j. quality_increases (σ j) (σ' j)"
  have "∀dip∈kD(rt (σ i)). nhop dip ≠ dip →
    dip∈kD(rt (σ' (nhop dip))) ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (nhop dip))) dip"
  proof (intro ballI impI)
    fix dip
    assume "dip∈kD(rt (σ i))"
    and "nhop dip ≠ dip"
    with pre have "dip∈kD(rt (σ (nhop dip)))"
    and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip"

```

```

    by auto
  from qinc have qinc_nhop: "quality_increases ( $\sigma$  (nhop dip)) ( $\sigma'$  (nhop dip))" ..
  with  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma \text{ (nhop dip)})) \rangle$  have "dip  $\in$  kD (rt ( $\sigma'$  (nhop dip)))" ..

  moreover have "nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
  proof -
    from  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma \text{ (nhop dip)})) \rangle$  qinc_nhop
      have "nsqn (rt ( $\sigma$  (nhop dip))) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip" ..
      with  $\langle \text{nsqn}(\text{rt } (\sigma \text{ i})) \text{ dip} \leq \text{nsqn}(\text{rt } (\sigma \text{ (nhop dip)})) \text{ dip} \rangle$  show ?thesis
      by simp
  qed

  ultimately show "dip  $\in$  kD (rt ( $\sigma'$  (nhop dip)))
     $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip" ..
  qed
} note basic = this

{ fix nhop and  $\sigma \sigma' :: \text{"ip} \Rightarrow \text{state"}$ 
  assume pre: " $\forall \text{dip} \in \text{kD}(\text{rt } (\sigma \text{ i})). \text{nhop dip} \neq \text{dip} \longrightarrow \text{dip} \in \text{kD}(\text{rt } (\sigma \text{ (nhop dip)}))$ 
     $\wedge$  nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma$  (nhop dip))) dip"
    and ndest: " $\forall \text{ripc} \in \text{dom}(\text{dests } (\sigma \text{ i})). \text{ripc} \in \text{kD}(\text{rt } (\sigma \text{ (sip } (\sigma \text{ i)})))$ 
     $\wedge$  the (dests ( $\sigma$  i) ripc) - 1  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) ripc"
    and issip: " $\forall \text{ip} \in \text{dom}(\text{dests } (\sigma \text{ i})). \text{nhop ip} = \text{sip } (\sigma \text{ i})$ "
    and qinc: " $\forall j. \text{quality\_increases } (\sigma \text{ j}) (\sigma' \text{ j})$ "
  have " $\forall \text{dip} \in \text{kD}(\text{rt } (\sigma \text{ i})). \text{nhop dip} \neq \text{dip} \longrightarrow \text{dip} \in \text{kD}(\text{rt } (\sigma' \text{ (nhop dip)}))$ 
     $\wedge$  nsqn (invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i))) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
  proof (intro ballI impI)
    fix dip
    assume "dip  $\in$  kD (rt ( $\sigma$  i))"
      and "nhop dip  $\neq$  dip"
    with pre and qinc have "dip  $\in$  kD (rt ( $\sigma'$  (nhop dip)))"
      and "nsqn (rt ( $\sigma$  i)) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
    by (auto dest!: basic)

    have "nsqn (invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i))) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
    proof (cases "dip  $\in$  dom (dests ( $\sigma$  i))")
      assume "dip  $\in$  dom (dests ( $\sigma$  i))"
      with  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma \text{ i})) \rangle$  obtain dsn where "dests ( $\sigma$  i) dip = Some dsn"
        by auto
      with  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma \text{ i})) \rangle$  have "nsqn (invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i))) dip = dsn - 1"
        by (rule nsqn_invalidate_eq)
      moreover have "dsn - 1  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
      proof -
        from  $\langle \text{dests } (\sigma \text{ i}) \text{ dip} = \text{Some dsn} \rangle$  have "the (dests ( $\sigma$  i) dip) = dsn" by simp
        with ndest and  $\langle \text{dip} \in \text{dom}(\text{dests } (\sigma \text{ i})) \rangle$  have "dip  $\in$  kD (rt ( $\sigma$  (sip ( $\sigma$  i))))"
          "dsn - 1  $\leq$  nsqn (rt ( $\sigma$  (sip ( $\sigma$  i)))) dip"
        by auto
        moreover from issip and  $\langle \text{dip} \in \text{dom}(\text{dests } (\sigma \text{ i})) \rangle$  have "nhop dip = sip ( $\sigma$  i)" ..
        ultimately have "dip  $\in$  kD (rt ( $\sigma$  (nhop dip)))"
          and "dsn - 1  $\leq$  nsqn (rt ( $\sigma$  (nhop dip))) dip" by auto
        with qinc show "dsn - 1  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip"
          by simp (metis kD_nsqn_quality_increases_trans)
      qed
    qed
    ultimately show ?thesis by simp
  next
    assume "dip  $\notin$  dom (dests ( $\sigma$  i))"
    with  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma \text{ i})) \rangle$ 
      have "nsqn (invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i))) dip = nsqn (rt ( $\sigma$  i)) dip"
        by (rule nsqn_invalidate_other)
      with  $\langle \text{nsqn}(\text{rt } (\sigma \text{ i})) \text{ dip} \leq \text{nsqn}(\text{rt } (\sigma' \text{ (nhop dip)})) \text{ dip} \rangle$  show ?thesis by simp
    qed
  with  $\langle \text{dip} \in \text{kD}(\text{rt } (\sigma' \text{ (nhop dip)})) \rangle$ 
    show "dip  $\in$  kD (rt ( $\sigma'$  (nhop dip)))
       $\wedge$  nsqn (invalidate (rt ( $\sigma$  i)) (dests ( $\sigma$  i))) dip  $\leq$  nsqn (rt ( $\sigma'$  (nhop dip))) dip" ..
}

```

```

qed
} note basic_prerr = this

{ fix  $\sigma \sigma' :: \text{"ip} \Rightarrow \text{state}"$ 
  assume a1: " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}$ 
     $\longrightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$ 
     $\wedge \text{nsqn}(\text{rt}(\sigma i)) \text{dip} \leq \text{nsqn}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))) \text{dip}"$ 
  and a2: " $\forall j. \text{quality\_increases}(\sigma j)(\sigma' j)"$ 
  have " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)).$ 
     $\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i))(\text{sip}(\sigma i))(0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}(\sigma i))) \text{dip}) \neq \text{dip} \longrightarrow$ 
     $\text{dip} \in \text{kD}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i))(\text{sip}(\sigma i))$ 
       $(0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}(\sigma i)))$ 
       $\text{dip})))) \wedge$ 
     $\text{nsqn}(\text{update}(\text{rt}(\sigma i))(\text{sip}(\sigma i))(0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}(\sigma i))) \text{dip}$ 
     $\leq \text{nsqn}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i))(\text{sip}(\sigma i))$ 
       $(0, \text{unk}, \text{val}, \text{Suc } 0, \text{sip}(\sigma i)))$ 
       $\text{dip}))))$ 
     $\text{dip}"$  (is " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). ?P \text{dip}"$ )
proof
  fix dip
  assume "dip  $\in \text{kD}(\text{rt}(\sigma i))"$ 
  with a1 and a2
    have " $\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip} \longrightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$ 
       $\wedge \text{nsqn}(\text{rt}(\sigma i)) \text{dip} \leq \text{nsqn}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))) \text{dip}"$ 
      by - (drule(1) basic, auto)
    thus "?P dip" by (cases "dip = sip( $\sigma i$ )") auto
qed
} note nhop_update_sip = this

{ fix  $\sigma \sigma' \text{ oip sip osn hops}$ 
  assume pre: " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}$ 
     $\longrightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$ 
     $\wedge \text{nsqn}(\text{rt}(\sigma i)) \text{dip} \leq \text{nsqn}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))) \text{dip}"$ 
  and qinc: " $\forall j. \text{quality\_increases}(\sigma j)(\sigma' j)"$ 
  and *: " $\text{sip} \neq \text{oip} \longrightarrow \text{oip} \in \text{kD}(\text{rt}(\sigma \text{sip}))$ 
     $\wedge \text{osn} \leq \text{nsqn}(\text{rt}(\sigma \text{sip})) \text{oip}$ 
     $\wedge (\text{nsqn}(\text{rt}(\sigma \text{sip})) \text{oip} = \text{osn}$ 
     $\longrightarrow \text{the}(\text{dhops}(\text{rt}(\sigma \text{sip})) \text{oip}) \leq \text{hops}$ 
     $\vee \text{the}(\text{flag}(\text{rt}(\sigma \text{sip})) \text{oip}) = \text{inv})"$ 
from pre and qinc
  have pre': " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}$ 
     $\longrightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$ 
     $\wedge \text{nsqn}(\text{rt}(\sigma i)) \text{dip} \leq \text{nsqn}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))) \text{dip}"$ 
  by (rule basic)
  have "( $\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip})) \text{oip}) \neq \text{oip}$ 
     $\longrightarrow \text{oip} \in \text{kD}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}$ 
       $(\text{osn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip})) \text{oip}))))$ 
     $\wedge \text{nsqn}(\text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip})) \text{oip}$ 
     $\leq \text{nsqn}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}$ 
       $(\text{osn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip})) \text{oip})))) \text{oip}"$ 
    (is "?nhop_not_oip  $\longrightarrow$  ?oip_in_kD  $\wedge$  ?nsqn_le_nsqn")
proof (rule, split update_rt_split_asm)
  assume "rt( $\sigma i$ ) = update( $\text{rt}(\sigma i)$ ) oip( $\text{osn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip})"$ 
  and " $\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{oip}) \neq \text{oip}"$ 
  with pre' show "?oip_in_kD  $\wedge$  ?nsqn_le_nsqn" by auto
next
  assume rtnot: " $\text{rt}(\sigma i) \neq \text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc } \text{hops}, \text{sip})"$ 
  and notoip: ?nhop_not_oip
  with * qinc have ?oip_in_kD
  by (clarsimp elim!: kD_quality_increases)
  moreover with * pre qinc rtnot notoip have ?nsqn_le_nsqn
  by simp (metis kD_nsqn_quality_increases_trans)
  ultimately show "?oip_in_kD  $\wedge$  ?nsqn_le_nsqn" ..
qed

```

```

} note update1 = this

{ fix  $\sigma$   $\sigma'$  oip sip osn hops
  assume pre: " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}$ 
     $\rightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$ 
     $\wedge \text{nsqn}(\text{rt}(\sigma i)) \text{dip} \leq \text{nsqn}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))) \text{dip}"$ 
  and qinc: " $\forall j. \text{quality\_increases}(\sigma j)(\sigma' j)"$ 
  and *: " $\text{sip} \neq \text{oip} \rightarrow \text{oip} \in \text{kD}(\text{rt}(\sigma \text{sip}))$ 
     $\wedge \text{osn} \leq \text{nsqn}(\text{rt}(\sigma \text{sip})) \text{oip}$ 
     $\wedge (\text{nsqn}(\text{rt}(\sigma \text{sip})) \text{oip} = \text{osn}$ 
     $\rightarrow \text{the}(\text{dhops}(\text{rt}(\sigma \text{sip})) \text{oip}) \leq \text{hops}$ 
     $\vee \text{the}(\text{flag}(\text{rt}(\sigma \text{sip})) \text{oip}) = \text{inv}"$ 

  from pre and qinc
  have pre': " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}$ 
     $\rightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$ 
     $\wedge \text{nsqn}(\text{rt}(\sigma i)) \text{dip} \leq \text{nsqn}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))) \text{dip}"$ 
  by (rule basic)
  have " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)).$ 
     $\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip})) \text{dip}) \neq \text{dip}$ 
     $\rightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}$ 
       $(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip})) \text{dip}))))$ 
     $\wedge \text{nsqn}(\text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip})) \text{dip}$ 
     $\leq \text{nsqn}(\text{rt}(\sigma'(\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}$ 
       $(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip})) \text{dip})))) \text{dip}"$ 
    (is " $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \_ \rightarrow ?\text{dip\_in\_kD} \text{dip} \wedge ?\text{nsqn\_le\_nsqn} \text{dip}"$ )
  proof (intro ballI impI, split update_rt_split_asm)
  fix dip
  assume " $\text{dip} \in \text{kD}(\text{rt}(\sigma i))"$ 
  and " $\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}"$ 
  and " $\text{rt}(\sigma i) = \text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip})"$ 
  with pre' show " $?\text{dip\_in\_kD} \text{dip} \wedge ?\text{nsqn\_le\_nsqn} \text{dip}"$  by simp
  next
  fix dip
  assume " $\text{dip} \in \text{kD}(\text{rt}(\sigma i))"$ 
  and notdip: " $\text{the}(\text{nhop}(\text{update}(\text{rt}(\sigma i)) \text{oip}$ 
     $(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip})) \text{dip}) \neq \text{dip}"$ 
  and rtnot: " $\text{rt}(\sigma i) \neq \text{update}(\text{rt}(\sigma i)) \text{oip}(\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip})"$ 
  show " $?\text{dip\_in\_kD} \text{dip} \wedge ?\text{nsqn\_le\_nsqn} \text{dip}"$ 
  proof (cases "dip = oip")
  assume "dip  $\neq$  oip"
  with pre'  $\langle \text{dip} \in \text{kD}(\text{rt}(\sigma i)) \rangle$  notdip
  show ?thesis by clarsimp
  next
  assume "dip = oip"
  with rtnot qinc  $\langle \text{dip} \in \text{kD}(\text{rt}(\sigma i)) \rangle$  notdip *
  have " $?\text{dip\_in\_kD} \text{dip}"$ 
  by simp (metis kD_quality_increases)
  moreover from  $\langle \text{dip} = \text{oip} \rangle$  rtnot qinc  $\langle \text{dip} \in \text{kD}(\text{rt}(\sigma i)) \rangle$  notdip *
  have " $?\text{nsqn\_le\_nsqn} \text{dip}"$  by simp (metis kD_nsqn_quality_increases_trans)
  ultimately show ?thesis ..
  qed
  qed
} note update2 = this

have "opaadv i  $\models$  ( $?S, ?U \rightarrow$ ) onl  $\Gamma_{AODV}(\lambda(\sigma, \_).$ 
   $\forall \text{dip} \in \text{kD}(\text{rt}(\sigma i)). \text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}) \neq \text{dip}$ 
   $\rightarrow \text{dip} \in \text{kD}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip}))))$ 
   $\wedge \text{nsqn}(\text{rt}(\sigma i)) \text{dip} \leq \text{nsqn}(\text{rt}(\sigma(\text{the}(\text{nhop}(\text{rt}(\sigma i)) \text{dip})))) \text{dip}"$ 
  by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aadv_wf
    oadv_trans]
    onl_oinvariant_sterms [OF aadv_wf odests_vD_inc_nsqn]
    onl_oinvariant_sterms [OF aadv_wf oprerr_guard]
    onl_oinvariant_sterms [OF aadv_wf rreq_sip]
    onl_oinvariant_sterms [OF aadv_wf rrep_sip]

```



```

        onl_oinvariant_sterms [OF aadv_wf rerr_sip]
        other_quality_increases
        other_localD
    solve: basic basic_prerr
    simp add: seqlsimp nsqn_invalidate nhop_update_sip
    simp del: One_nat_def
(rule conjI, erule(2) update1, erule(2) update2)+

```

thus ?thesis unfolding Let\_def by auto  
qed

Proposition 7.30

```

lemmas okD_unk_or_atleast_one =
  open_seq_invariant [OF kD_unk_or_atleast_one initiali_aadv,
    simplified seql_onl_swap]

```

```

lemmas ozero_seq_unk_hops_one =
  open_seq_invariant [OF zero_seq_unk_hops_one initiali_aadv,
    simplified seql_onl_swap]

```

lemma oreachable\_fresh\_okD\_unk\_or\_atleast\_one:

```

  fixes dip
  assumes "(σ, p) ∈ oreachable (opaadv i)
    (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
      ∧ msg_zhops m)))
    (other quality_increases {i})"
  and "dip ∈ kD(rt (σ i))"
  shows "π3(the (rt (σ i) dip)) = unk ∨ 1 ≤ π2(the (rt (σ i) dip))"
  (is "?P dip")

```

proof -

```

  have "∃l. l ∈ labels ΓAADV p" by (metis aadv_ex_label)
  with assms(1) have "∀dip ∈ kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaadv_trans aadv_trans]],
    auto dest!: otherwith_actionD onlD simp: seqlsimp)
  with ⟨dip ∈ kD(rt (σ i))⟩ show ?thesis by simp
qed

```

lemma oreachable\_fresh\_ozero\_seq\_unk\_hops\_one:

```

  fixes dip
  assumes "(σ, p) ∈ oreachable (opaadv i)
    (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
      ∧ msg_zhops m)))
    (other quality_increases {i})"
  and "dip ∈ kD(rt (σ i))"
  shows "sqn (rt (σ i) dip) = 0 →
    sqnf (rt (σ i) dip) = unk
    ∧ the (dhops (rt (σ i) dip)) = 1
    ∧ the (nhop (rt (σ i) dip)) = dip"
  (is "?P dip")

```

proof -

```

  have "∃l. l ∈ labels ΓAADV p" by (metis aadv_ex_label)
  with assms(1) have "∀dip ∈ kD (rt (σ i)). ?P dip"
  by - (drule oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaadv_trans aadv_trans]],
    auto dest!: onlD otherwith_actionD simp: seqlsimp)
  with ⟨dip ∈ kD(rt (σ i))⟩ show ?thesis by simp
qed

```

lemma seq\_nhop\_quality\_increases':

```

  shows "opaadv i ⊨ (otherwith (op=) {i}
    (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} →)
    onl ΓAADV (λ(σ, _). ∀dip. let nhip = the (nhop (rt (σ i) dip))
      in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip))
      ∧ nhip ≠ dip)

```

→ (rt (σ i)) ⊆<sub>dip</sub> (rt (σ nhop)))"

(is "\_ ⊨ (?S i, \_ →) \_")

proof -

have weaken:

" $\bigwedge_p I Q R P. p \models (\text{otherwith quality\_increases } I (\text{orecvmsg } Q), \text{ other quality\_increases } I \rightarrow) P$   
 $\implies p \models (\text{otherwith } (\text{op}=) I (\text{orecvmsg } (\lambda\sigma m. Q \sigma m \wedge R \sigma m)), \text{ other quality\_increases } I \rightarrow) P$ "  
 by auto

{

fix i a and σ σ' :: "ip ⇒ state"

assume a1: "∀dip. dip ∈ vD(rt (σ i))  
 ∧ dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))  
 ∧ (the (nhop (rt (σ i)) dip)) ≠ dip  
 → rt (σ i) ⊆<sub>dip</sub> rt (σ (the (nhop (rt (σ i)) dip)))"

and ow: "?S i σ σ' a"

have "∀dip. dip ∈ vD(rt (σ i))  
 ∧ dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))  
 ∧ (the (nhop (rt (σ i)) dip)) ≠ dip  
 → rt (σ i) ⊆<sub>dip</sub> rt (σ' (the (nhop (rt (σ i)) dip)))"

proof clarify

fix dip

assume a2: "dip ∈ vD(rt (σ i))"  
 and a3: "dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))"  
 and a4: "(the (nhop (rt (σ i)) dip)) ≠ dip"

from ow have "∀j. j ≠ i → σ j = σ' j" by auto

show "rt (σ i) ⊆<sub>dip</sub> rt (σ' (the (nhop (rt (σ i)) dip)))"

proof (cases "(the (nhop (rt (σ i)) dip)) = i")

assume "(the (nhop (rt (σ i)) dip)) = i"

with ⟨dip ∈ vD(rt (σ i))⟩ have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))" by simp

with a1 a2 a4 have "rt (σ i) ⊆<sub>dip</sub> rt (σ (the (nhop (rt (σ i)) dip)))" by simp

with ⟨(the (nhop (rt (σ i)) dip)) = i⟩ have "rt (σ i) ⊆<sub>dip</sub> rt (σ i)" by simp

hence False by simp

thus ?thesis ..

next

assume "(the (nhop (rt (σ i)) dip)) ≠ i"

with ⟨∀j. j ≠ i → σ j = σ' j⟩

have \*: "σ (the (nhop (rt (σ i)) dip)) = σ' (the (nhop (rt (σ i)) dip))" by simp

with ⟨dip ∈ vD (rt (σ' (the (nhop (rt (σ i)) dip))))⟩

have "dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))" by simp

with a1 a2 a4 have "rt (σ i) ⊆<sub>dip</sub> rt (σ (the (nhop (rt (σ i)) dip)))" by simp

with \* show ?thesis by simp

qed

qed

} note basic = this

{ fix σ σ' a dip sip i

assume a1: "∀dip. dip ∈ vD(rt (σ i))  
 ∧ dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))  
 ∧ the (nhop (rt (σ i)) dip) ≠ dip  
 → rt (σ i) ⊆<sub>dip</sub> rt (σ (the (nhop (rt (σ i)) dip)))"

and ow: "?S i σ σ' a"

have "∀dip. dip ∈ vD(update (rt (σ i)) sip (0, unk, val, Suc 0, sip))  
 ∧ dip ∈ vD(rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip))))  
 ∧ the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip) ≠ dip  
 → update (rt (σ i)) sip (0, unk, val, Suc 0, sip)  
 ⊆<sub>dip</sub> rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip)))"

proof clarify

fix dip

assume a2: "dip ∈ vD (update (rt (σ i)) sip (0, unk, val, Suc 0, sip))"

and a3: "dip ∈ vD(rt (σ' (the (nhop  
 (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip))))"

and a4: "the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip) ≠ dip"

show "update (rt (σ i)) sip (0, unk, val, Suc 0, sip)

⊆<sub>dip</sub> rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip)))"

proof (cases "dip = sip")

```

    assume "dip = sip"
    with ⟨the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip)) dip) ≠ dip⟩
    have False by simp
    thus ?thesis ..
  next
    assume [simp]: "dip ≠ sip"
    from a2 have "dip ∈ vD(rt (σ i)) ∨ dip = sip"
      by (rule vD_update_val)
    with ⟨dip ≠ sip⟩ have "dip ∈ vD(rt (σ i))" by simp
    moreover from a3 have "dip ∈ vD(rt (σ' (the (nhop (rt (σ i)) dip))))" by simp
    moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
    ultimately have "rt (σ i) ⊑dip rt (σ' (the (nhop (rt (σ i)) dip)))"
      using a1 ow by - (drule(1) basic, simp)
    with ⟨dip ≠ sip⟩ show ?thesis
      by - (erule rt_strictly_fresher_update_other, simp)
  qed
qed
} note update_0_unk = this

{ fix σ a σ' nhop
  assume pre: "∀dip. dip ∈ vD(rt (σ i)) ∧ dip ∈ vD(rt (σ (nhop dip))) ∧ nhop dip ≠ dip
    → rt (σ i) ⊑dip rt (σ (nhop dip))"
    and ow: "?S i σ σ' a"
  have "∀dip. dip ∈ vD (invalidate (rt (σ i)) (dests (σ i)))
    ∧ dip ∈ vD (rt (σ' (nhop dip))) ∧ nhop dip ≠ dip
    → rt (σ i) ⊑dip rt (σ' (nhop dip))"
  proof clarify
    fix dip
    assume "dip ∈ vD(invalidate (rt (σ i)) (dests (σ i)))"
      and "dip ∈ vD(rt (σ' (nhop dip)))"
      and "nhop dip ≠ dip"
    from this(1) have "dip ∈ vD (rt (σ i))"
      by (clarsimp dest!: vD_invalidate_vD_not_dests)
    moreover from ow have "∀j. j ≠ i → σ j = σ' j" by auto
    ultimately have "rt (σ i) ⊑dip rt (σ (nhop dip))"
      using pre ⟨dip ∈ vD (rt (σ' (nhop dip)))⟩ ⟨nhop dip ≠ dip⟩
      by metis
    with ⟨∀j. j ≠ i → σ j = σ' j⟩ show "rt (σ i) ⊑dip rt (σ' (nhop dip))"
      by (metis rt_strictly_fresher_irefl)
  qed
} note invalidate = this

{ fix σ a σ' dip oip osn sip hops i
  assume pre: "∀dip. dip ∈ vD (rt (σ i))
    ∧ dip ∈ vD (rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) ⊑dip rt (σ (the (nhop (rt (σ i)) dip)))"
    and ow: "?S i σ σ' a"
    and "Suc 0 ≤ osn"
    and a6: "sip ≠ oip → oip ∈ kD (rt (σ sip))
      ∧ osn ≤ nsqn (rt (σ sip)) oip
      ∧ (nsqn (rt (σ sip)) oip = osn
        → the (dhops (rt (σ sip)) oip) ≤ hops
          ∨ the (flag (rt (σ sip)) oip) = inv)"
  and after: "σ' i = σ i (|rt := update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)|)"
  have "∀dip. dip ∈ vD (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip))
    ∧ dip ∈ vD (rt (σ' (the (nhop (update (rt (σ i)) oip
      (osn, kno, val, Suc hops, sip)) dip))))
    ∧ the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip) ≠ dip
    → update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)
      ⊑dip
      rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip)))"
  proof clarify
    fix dip

```

```

assume a2: "dip∈vD(update (rt (σ i)) oip (osn, kno, val, Suc (hops), sip))"
  and a3: "dip∈vD(rt (σ' (the (nhop (update (rt (σ i)) oip
    (osn, kno, val, Suc hops, sip)) dip))))"
  and a4: "the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip) ≠ dip"
from ow have a5: "∀j. j ≠ i → σ j = σ' j" by auto
show "update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)
  ⊆dip
  rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) dip)))"
(is "?rt1 ⊆dip ?rt2 dip")
proof (cases "?rt1 = rt (σ i)")
  assume nochange [simp]:
    "update (rt (σ i)) oip (osn, kno, val, Suc hops, sip) = rt (σ i)"

from after have "σ' i = σ i" by simp
with a5 have "∀j. σ j = σ' j" by metis

from a2 have "dip∈vD (rt (σ i))" by simp
moreover from a3 have "dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))"
  using nochange and ⟨∀j. σ j = σ' j⟩ by clarsimp
moreover from a4 have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
ultimately have "rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))"
  using pre by simp

hence "rt (σ i) ⊆dip rt (σ' (the (nhop (rt (σ i)) dip)))"
  using ⟨∀j. σ j = σ' j⟩ by simp
thus "?thesis" by simp
next
assume change: "?rt1 ≠ rt (σ i)"
from after a2 have "dip∈kD(rt (σ' i))" by auto
show ?thesis
proof (cases "dip = oip")
  assume "dip ≠ oip"

  with a2 have "dip∈vD (rt (σ i))" by auto
  moreover with a3 a5 after and ⟨dip ≠ oip⟩
  have "dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))"
    by simp metis
  moreover from a4 and ⟨dip ≠ oip⟩ have "the (nhop (rt (σ i)) dip) ≠ dip" by simp
  ultimately have "rt (σ i) ⊆dip rt (σ (the (nhop (rt (σ i)) dip)))"
    using pre by simp

  with after and a5 and ⟨dip ≠ oip⟩ show ?thesis
    by simp (metis rt_strictly_fresher_update_other
      rt_strictly_fresher_irefl)
next
assume "dip = oip"

with a4 and change have "sip ≠ oip" by simp
with a6 have "oip∈kD(rt (σ sip))"
  and "osn ≤ nsqn (rt (σ sip)) oip" by auto

from a3 change ⟨dip = oip⟩ have "oip∈vD(rt (σ' sip))" by simp
hence "the (flag (rt (σ' sip)) oip) = val" by simp

from ⟨oip∈kD(rt (σ sip))⟩
have "osn < nsqn (rt (σ' sip)) oip ∨ (osn = nsqn (rt (σ' sip)) oip
  ∧ the (dhops (rt (σ' sip)) oip) ≤ hops)"
proof
  assume "oip∈vD(rt (σ sip))"
  hence "the (flag (rt (σ sip)) oip) = val" by simp
  with a6 ⟨sip ≠ oip⟩ have "nsqn (rt (σ sip)) oip = osn →
    the (dhops (rt (σ sip)) oip) ≤ hops"
    by simp
  show ?thesis

```

```

proof (cases "sip = i")
  assume "sip ≠ i"
  with a5 have "σ sip = σ' sip" by simp
  with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
    and ⟨nsqn (rt (σ sip)) oip = osn ⟶ the (dhops (rt (σ sip)) oip) ≤ hops⟩
  show ?thesis by auto
next
— alternative to using sip_not_ip
assume [simp]: "sip = i"
have "?rt1 = rt (σ i)"
proof (rule update_cases_kD, simp_all)
  from ⟨Suc 0 ≤ osn⟩ show "0 < osn" by simp
next
  from ⟨oip ∈ kD(rt (σ sip))⟩ and ⟨sip = i⟩ show "oip ∈ kD(rt (σ i))"
  by simp
next
  assume "sqn (rt (σ i)) oip < osn"
  also from ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
    have "... ≤ nsqn (rt (σ i)) oip" by simp
  also have "... ≤ sqn (rt (σ i)) oip"
  by (rule nsqn_sqn)
  finally have "sqn (rt (σ i)) oip < sqn (rt (σ i)) oip" .
  hence False by simp
  thus "(λa. if a = oip
    then Some (osn, kno, val, Suc hops, i)
    else rt (σ i) a) = rt (σ i)" ..
next
  assume "sqn (rt (σ i)) oip = osn"
  and "Suc hops < the (dhops (rt (σ i)) oip)"
  from this(1) and ⟨oip ∈ vD (rt (σ sip))⟩ have "nsqn (rt (σ i)) oip = osn"
  by simp
  with ⟨nsqn (rt (σ sip)) oip = osn ⟶ the (dhops (rt (σ sip)) oip) ≤ hops⟩
    have "the (dhops (rt (σ i)) oip) ≤ hops" by simp
  with ⟨Suc hops < the (dhops (rt (σ i)) oip)⟩ have False by simp
  thus "(λa. if a = oip
    then Some (osn, kno, val, Suc hops, i)
    else rt (σ i) a) = rt (σ i)" ..
next
  assume "the (flag (rt (σ i)) oip) = inv"
  with ⟨the (flag (rt (σ sip)) oip) = val⟩ have False by simp
  thus "(λa. if a = oip
    then Some (osn, kno, val, Suc hops, i)
    else rt (σ i) a) = rt (σ i)" ..
next
  from ⟨oip ∈ kD(rt (σ sip))⟩
    show "(λa. if a = oip then Some (the (rt (σ i) oip)) else rt (σ i) a) = rt (σ i)"
  by (auto dest!: kD_Some)
qed
with change have False ..
thus ?thesis ..
qed
next
assume "oip ∈ iD(rt (σ sip))"
with ⟨the (flag (rt (σ' sip)) oip) = val⟩ and a5 have "sip = i"
  by (metis f.distinct(1) iD_flag_is_inv)
from ⟨oip ∈ iD(rt (σ sip))⟩ have "the (flag (rt (σ sip)) oip) = inv" by auto
with ⟨sip = i⟩ ⟨Suc 0 ≤ osn⟩ change after ⟨oip ∈ kD(rt (σ sip))⟩
  have "nsqn (rt (σ sip)) oip < nsqn (rt (σ' sip)) oip"
  unfolding update_def
  by (clarsimp split: option.split_asm split_if_asm)
  (auto simp: sqn_def)
with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩ have "osn < nsqn (rt (σ' sip)) oip"
  by simp
thus ?thesis ..

```

```

qed
thus ?thesis
proof
  assume osnlt: "osn < nsqn (rt (σ' sip)) oip"
  from ⟨dip ∈ kD(rt (σ' i))⟩ and ⟨dip = oip⟩ have "dip ∈ kD (?rt1)" by simp
  moreover from a3 have "dip ∈ kD(?rt2 dip)" by simp
  moreover have "nsqn ?rt1 dip < nsqn (?rt2 dip) dip"
  proof -
    have "nsqn ?rt1 oip = osn"
    by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
    also have "... < nsqn (rt (σ' sip)) oip" using osnlt .
    also have "... = nsqn (?rt2 oip) oip" by (simp add: change)
    finally show ?thesis
    using ⟨dip = oip⟩ by simp
  qed
  ultimately show ?thesis
  by (rule rt_strictly_fresher_ltI)
next
assume osneq: "osn = nsqn (rt (σ' sip)) oip ∧ the (dhops (rt (σ' sip)) oip) ≤ hops"

have "oip ∈ kD(?rt1)" by simp
moreover from a3 ⟨dip = oip⟩ have "oip ∈ kD(?rt2 oip)" by simp

moreover have "nsqn ?rt1 oip = nsqn (?rt2 oip) oip"
proof -
  from osneq have "osn = nsqn (rt (σ' sip)) oip" ..
  also have "osn = nsqn ?rt1 oip"
  by (simp add: ⟨dip = oip⟩ nsqn_update_changed_kno_val [OF change [THEN not_sym]])
  also have "nsqn (rt (σ' sip)) oip = nsqn (?rt2 oip) oip"
  by (simp add: change)
  finally show ?thesis .
qed

moreover have "π5(the (?rt2 oip oip)) < π5(the (?rt1 oip))"
proof -
  from osneq have "the (dhops (rt (σ' sip)) oip) ≤ hops" ..
  moreover from ⟨oip ∈ vD (rt (σ' sip))⟩ have "oip ∈ kD(rt (σ' sip))" by auto
  ultimately have "π5(the (rt (σ' sip) oip)) ≤ hops"
  by (auto simp add: proj5_eq_dhops)
  also from change after have "hops < π5(the (rt (σ' i) oip))"
  by (simp add: proj5_eq_dhops) (metis dhops_update_changed lessI)
  finally have "π5(the (rt (σ' sip) oip)) < π5(the (rt (σ' i) oip))" .
  with change after show ?thesis by simp
qed

ultimately have "?rt1 □oip ?rt2 oip"
  by (rule rt_strictly_fresher_eqI)
with ⟨dip = oip⟩ show ?thesis by simp
qed

qed
qed
qed
} note rreq_rrep_update = this

have "opaadv i ⊨ (otherwith (op=) {i} (orecvmsg (λσ m. msg_fresh σ m
                                                ∧ msg_zhops m))),
      other quality_increases {i} →)
  onl ΓAODV
  (λ(σ, _). ∀dip. dip ∈ vD (rt (σ i)) ∩ vD (rt (σ (the (nhop (rt (σ i)) dip))))
    ∧ the (nhop (rt (σ i)) dip) ≠ dip
    → rt (σ i) □dip rt (σ (the (nhop (rt (σ i)) dip))))"
proof (inv_cterms inv add: onl_oinvariant_sterms [OF aadv_wf rreq_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf rrep_sip [THEN weaken]]
  onl_oinvariant_sterms [OF aadv_wf rerr_sip [THEN weaken]])

```

```

        onl_oinvariant_sterms [OF aadv_wf oosn_rreq [THEN weaken]]
        onl_oinvariant_sterms [OF aadv_wf odsn_rreq [THEN weaken]]
    solve: basic update_0_unk invalidate rreq_rrep_update
    simp add: seqlsimp)
fix  $\sigma \sigma' p l$ 
assume or: " $(\sigma, p) \in \text{oreachable} (\text{opaadv } i) (?S \ i) \ (\text{other quality\_increases } \{i\})$ "
  and "other quality\_increases  $\{i\} \ \sigma \ \sigma'$ "
  and ll: " $l \in \text{labels } \Gamma_{AODV} \ p$ "
  and pre: " $\forall \text{dip. dip} \in vD \ (\text{rt } (\sigma \ i))$ 
     $\wedge \text{dip} \in vD(\text{rt } (\sigma \ (\text{the } (\text{nhop } (\text{rt } (\sigma \ i)) \ \text{dip}))))$ 
     $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma \ i)) \ \text{dip}) \neq \text{dip}$ 
     $\longrightarrow \text{rt } (\sigma \ i) \sqsubset_{\text{dip}} \text{rt } (\sigma \ (\text{the } (\text{nhop } (\text{rt } (\sigma \ i)) \ \text{dip})))$ "
from this(1-2)
  have or': " $(\sigma', p) \in \text{oreachable} (\text{opaadv } i) (?S \ i) \ (\text{other quality\_increases } \{i\})$ "
    by - (rule oreachable_other')

from or and ll have next_hop: " $\forall \text{dip. let } \text{nhip} = \text{the } (\text{nhop } (\text{rt } (\sigma \ i)) \ \text{dip})$ 
  in  $\text{dip} \in kD(\text{rt } (\sigma \ i)) \wedge \text{nhip} \neq \text{dip}$ 
   $\longrightarrow \text{dip} \in kD(\text{rt } (\sigma \ \text{nhip}))$ 
   $\wedge \text{nsqn } (\text{rt } (\sigma \ i)) \ \text{dip} \leq \text{nsqn } (\text{rt } (\sigma \ \text{nhip})) \ \text{dip}$ "
  by (auto dest!: onl_oinvariant_weakenD [OF seq_compare_next_hop'])

from or and ll have unk_hops_one: " $\forall \text{dip} \in kD \ (\text{rt } (\sigma \ i)). \ \text{sqn } (\text{rt } (\sigma \ i)) \ \text{dip} = 0$ 
   $\longrightarrow \text{sqnf } (\text{rt } (\sigma \ i)) \ \text{dip} = \text{unk}$ 
   $\wedge \text{the } (\text{dhops } (\text{rt } (\sigma \ i)) \ \text{dip}) = 1$ 
   $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma \ i)) \ \text{dip}) = \text{dip}$ "
  by (auto dest!: onl_oinvariant_weakenD [OF ozero_seq_unk_hops_one
    [OF oaadv_trans aadv_trans]]
    otherwith_actionD
    simp: seqlsimp)

from (other quality\_increases  $\{i\} \ \sigma \ \sigma'$ ) have " $\sigma' \ i = \sigma \ i$ " by auto
hence "quality\_increases  $(\sigma \ i) \ (\sigma' \ i)$ " by auto
with (other quality\_increases  $\{i\} \ \sigma \ \sigma'$ ) have " $\forall j. \ \text{quality\_increases } (\sigma \ j) \ (\sigma' \ j)$ "
  by - (erule otherE, metis singleton_iff)

show " $\forall \text{dip. dip} \in vD \ (\text{rt } (\sigma' \ i))$ 
   $\wedge \text{dip} \in vD \ (\text{rt } (\sigma' \ (\text{the } (\text{nhop } (\text{rt } (\sigma' \ i)) \ \text{dip}))))$ 
   $\wedge \text{the } (\text{nhop } (\text{rt } (\sigma' \ i)) \ \text{dip}) \neq \text{dip}$ 
   $\longrightarrow \text{rt } (\sigma' \ i) \sqsubset_{\text{dip}} \text{rt } (\sigma' \ (\text{the } (\text{nhop } (\text{rt } (\sigma' \ i)) \ \text{dip})))$ "
proof clarify
  fix dip
  assume "dip  $\in vD(\text{rt } (\sigma' \ i))$ "
  and "dip  $\in vD(\text{rt } (\sigma' \ (\text{the } (\text{nhop } (\text{rt } (\sigma' \ i)) \ \text{dip}))))$ "
  and "the (nhop (rt ( $\sigma' \ i$ )) dip)  $\neq$  dip"
  from this(1) and ( $\sigma' \ i = \sigma \ i$ ) have "dip  $\in vD(\text{rt } (\sigma \ i))$ "
    and "dip  $\in kD(\text{rt } (\sigma \ i))$ "
  by auto

from (the (nhop (rt ( $\sigma' \ i$ )) dip)  $\neq$  dip) and ( $\sigma' \ i = \sigma \ i$ )
  have "the (nhop (rt ( $\sigma \ i$ )) dip)  $\neq$  dip" (is "?nhip  $\neq$  _") by simp
with (dip  $\in kD(\text{rt } (\sigma \ i))$ ) and next_hop
  have "dip  $\in kD(\text{rt } (\sigma \ (?nhip)))$ "
  and nsqns: "nsqn (rt ( $\sigma \ i$ )) dip  $\leq$  nsqn (rt ( $\sigma \ ?nhip$ )) dip"
  by (auto simp: Let_def)

have " $0 < \text{sqn } (\text{rt } (\sigma \ i)) \ \text{dip}$ "
  proof (rule neq0_conv [THEN iffD1, OF notI])
    assume "sqn (rt ( $\sigma \ i$ )) dip = 0"
    with (dip  $\in kD(\text{rt } (\sigma \ i))$ ) and unk_hops_one
      have "?nhip = dip" by simp
    with (?nhip  $\neq$  dip) show False ..
  qed
also have "... = nsqn (rt ( $\sigma \ i$ )) dip"

```

```

  by (rule vD_nsqn_sqn [OF ⟨dip∈vD(rt (σ i))⟩, THEN sym])
  also have "... ≤ nsqn (rt (σ ?nhip)) dip"
  by (rule nsqns)
  also have "... ≤ sqn (rt (σ ?nhip)) dip"
  by (rule nsqn_sqn)
  finally have "0 < sqn (rt (σ ?nhip)) dip" .

```

```

have "rt (σ i) ⊑dip rt (σ' ?nhip)"
proof (cases "dip∈vD(rt (σ ?nhip))")
  assume "dip∈vD(rt (σ ?nhip))"
  with pre ⟨dip∈vD(rt (σ i))⟩ and ⟨?nhip ≠ dip⟩
  have "rt (σ i) ⊑dip rt (σ ?nhip)" by auto
  moreover from ⟨∀j. quality_increases (σ j) (σ' j)⟩
  have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
  ultimately show ?thesis
  using ⟨dip∈kD(rt (σ ?nhip))⟩
  by (rule strictly_fresher_quality_increases_right)

```

```

next
  assume "dip∉vD(rt (σ ?nhip))"
  with ⟨dip∈kD(rt (σ ?nhip))⟩ have "dip∈iD(rt (σ ?nhip))" ..
  hence "the (flag (rt (σ ?nhip)) dip) = inv"
  by auto
  have "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ ?nhip)) dip"
  by (rule nsqns)
  also from ⟨dip∈iD(rt (σ ?nhip))⟩
  have "... = sqn (rt (σ ?nhip)) dip - 1" ..
  also have "... < sqn (rt (σ' ?nhip)) dip"
  proof -
    from ⟨∀j. quality_increases (σ j) (σ' j)⟩
    have "quality_increases (σ ?nhip) (σ' ?nhip)" ..
    hence "∀ip. sqn (rt (σ ?nhip)) ip ≤ sqn (rt (σ' ?nhip)) ip" by auto
    hence "sqn (rt (σ ?nhip)) dip ≤ sqn (rt (σ' ?nhip)) dip" ..
    with ⟨0 < sqn (rt (σ ?nhip)) dip⟩ show ?thesis by auto
  qed
  also have "... = nsqn (rt (σ' ?nhip)) dip"
  proof (rule vD_nsqn_sqn [THEN sym])
    from ⟨dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and ⟨σ' i = σ i⟩
    show "dip∈vD(rt (σ' ?nhip))" by simp
  qed
  finally have "nsqn (rt (σ i)) dip < nsqn (rt (σ' ?nhip)) dip" .

```

```

moreover from ⟨dip∈vD(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and ⟨σ' i = σ i⟩
  have "dip∈kD(rt (σ' ?nhip))" by auto
  ultimately show "rt (σ i) ⊑dip rt (σ' ?nhip)"
  using ⟨dip∈kD(rt (σ i))⟩ by - (rule rt_strictly_fresher_ltI)

```

```

qed
with ⟨σ' i = σ i⟩ show "rt (σ' i) ⊑dip rt (σ' (the (nhop (rt (σ' i)) dip)))"
  by simp
qed
thus ?thesis unfolding Let_def .
qed

```

lemma seq\_compare\_next\_hop:

```

fixes w
shows "opaodv i ⊨ (otherwith (op=) {i} (orecvmsg msg_fresh),
  other quality_increases {i} →)
  global (λσ. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ kD(rt (σ i)) ∧ nhip ≠ dip →
      dip ∈ kD(rt (σ nhip))
      ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ nhip)) dip)"
  by (rule oinvariant_weakenE [OF seq_compare_next_hop']) (auto dest!: onID)

```

lemma seq\_nhop\_quality\_increases:



```

shows "opaadv i  $\models$  (otherwith (op=) {i}
      (orecvmsg ( $\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m$ )),
      other quality_increases {i}  $\rightarrow$ )
      global ( $\lambda\sigma. \forall \text{dip. let } \text{nhip} = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip})
              \text{ in } \text{dip} \in \text{vD } (\text{rt } (\sigma i)) \cap \text{vD } (\text{rt } (\sigma \text{nhip})) \wedge \text{nhip} \neq \text{dip}
              \rightarrow (\text{rt } (\sigma i)) \sqsubset_{\text{dip}} (\text{rt } (\sigma \text{nhip}))$ )"
by (rule oinvariant_weakenE [OF seq_nhop_quality_increases']) (auto dest!: onlD)

```

end

## 5.10 Routing graphs and loop freedom

```

theory E_Loop_Freedom
imports E_Aadv_Predicates E_Fresher
begin

```

Define the central theorem that relates an invariant over network states to the absence of loops in the associate routing graph.

**definition**

```

rt_graph :: "(ip  $\Rightarrow$  state)  $\Rightarrow$  ip  $\Rightarrow$  ip rel"
where
  "rt_graph  $\sigma = (\lambda \text{dip.}
    \{(ip, ip') \mid ip \text{ ip'} \text{ dsn dsk hops.}
      ip \neq \text{dip} \wedge \text{rt } (\sigma ip) \text{ dip} = \text{Some } (\text{dsn, dsk, val, hops, ip'})\})"$ 

```

Given the state of a network  $\sigma$ , a routing graph for a given destination  $\text{dip}$  abstracts the details of routing tables into nodes (ip addresses) and vertices (valid routes between ip addresses).

**lemma** *rt\_graphE* [elim]:

```

  fixes n dip ip ip'
  assumes "(ip, ip')  $\in$  rt_graph  $\sigma$  dip"
  shows "ip  $\neq$  dip  $\wedge$  ( $\exists r. \text{rt } (\sigma ip) = r
    \wedge (\exists \text{dsn dsk hops. } r \text{ dip} = \text{Some } (\text{dsn, dsk, val, hops, ip'})$ )"
using assms unfolding rt_graph_def by auto

```

**lemma** *rt\_graph\_vD* [dest]:

```

" $\bigwedge ip \text{ ip'} \sigma \text{ dip. } (ip, ip') \in \text{rt\_graph } \sigma \text{ dip} \implies \text{dip} \in \text{vD}(\text{rt } (\sigma ip))"$ 
unfolding rt_graph_def vD_def by auto

```

**lemma** *rt\_graph\_vD\_trans* [dest]:

```

" $\bigwedge ip \text{ ip'} \sigma \text{ dip. } (ip, ip') \in (\text{rt\_graph } \sigma \text{ dip})^+ \implies \text{dip} \in \text{vD}(\text{rt } (\sigma ip))"$ 
by (erule converse_tranclE) auto

```

**lemma** *rt\_graph\_not\_dip* [dest]:

```

" $\bigwedge ip \text{ ip'} \sigma \text{ dip. } (ip, ip') \in \text{rt\_graph } \sigma \text{ dip} \implies ip \neq \text{dip}"$ 
unfolding rt_graph_def by auto

```

**lemma** *rt\_graph\_not\_dip\_trans* [dest]:

```

" $\bigwedge ip \text{ ip'} \sigma \text{ dip. } (ip, ip') \in (\text{rt\_graph } \sigma \text{ dip})^+ \implies ip \neq \text{dip}"$ 
by (erule converse_tranclE) auto

```

NB: the property below cannot be lifted to the transitive closure

**lemma** *rt\_graph\_nhip\_is\_nhop* [dest]:

```

" $\bigwedge ip \text{ ip'} \sigma \text{ dip. } (ip, ip') \in \text{rt\_graph } \sigma \text{ dip} \implies ip' = \text{the } (\text{nhop } (\text{rt } (\sigma ip)) \text{ dip})"$ 
unfolding rt_graph_def by auto

```

**theorem** *inv\_to\_loop\_freedom*:

```

  assumes " $\forall i \text{ dip. let } \text{nhip} = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip})
    \text{ in } \text{dip} \in \text{vD } (\text{rt } (\sigma i)) \cap \text{vD } (\text{rt } (\sigma \text{nhip})) \wedge \text{nhip} \neq \text{dip}
    \rightarrow (\text{rt } (\sigma i)) \sqsubset_{\text{dip}} (\text{rt } (\sigma \text{nhip}))"$ 
  shows " $\forall \text{dip. irrefl } ((\text{rt\_graph } \sigma \text{ dip})^+)"$ 
using assms proof (intro allI)
  fix  $\sigma :: "ip \Rightarrow \text{state}"$  and dip
  assume inv: " $\forall ip \text{ dip.}$ "

```

```

      let nhip = the (nhop (rt (σ ip)) dip)
      in dip ∈ vD(rt (σ ip)) ∩ vD(rt (σ nhip)) ∧
         nhip ≠ dip → rt (σ ip) ⊑dip rt (σ nhip)"
{ fix ip ip'
  assume "(ip, ip') ∈ (rt_graph σ dip) +"
    and "dip ∈ vD(rt (σ ip'))"
    and "ip' ≠ dip"
  hence "rt (σ ip) ⊑dip rt (σ ip')"
  proof induction
    fix nhip
    assume "(ip, nhip) ∈ rt_graph σ dip"
      and "dip ∈ vD(rt (σ nhip))"
      and "nhip ≠ dip"
    from ⟨(ip, nhip) ∈ rt_graph σ dip⟩ have "dip ∈ vD(rt (σ ip))"
      and "nhip = the (nhop (rt (σ ip)) dip)"

      by auto
    from ⟨dip ∈ vD(rt (σ ip))⟩ and ⟨dip ∈ vD(rt (σ nhip))⟩
      have "dip ∈ vD(rt (σ ip)) ∩ vD(rt (σ nhip))" ..
    with ⟨nhip = the (nhop (rt (σ ip)) dip)⟩
      and ⟨nhip ≠ dip⟩
      and inv
      show "rt (σ ip) ⊑dip rt (σ nhip)"
      by (clarsimp simp: Let_def)
  next
    fix nhip nhip'
    assume "(ip, nhip) ∈ (rt_graph σ dip) +"
      and "(nhip, nhip') ∈ rt_graph σ dip"
      and IH: "[| dip ∈ vD(rt (σ nhip)); nhip ≠ dip |] ⇒ rt (σ ip) ⊑dip rt (σ nhip)"
      and "dip ∈ vD(rt (σ nhip'))"
      and "nhip' ≠ dip"
    from ⟨(nhip, nhip') ∈ rt_graph σ dip⟩ have 1: "dip ∈ vD(rt (σ nhip))"
      and 2: "nhip ≠ dip"
      and "nhip' = the (nhop (rt (σ nhip)) dip)"

      by auto
    from 1 2 have "rt (σ ip) ⊑dip rt (σ nhip)" by (rule IH)
    also have "rt (σ nhip) ⊑dip rt (σ nhip')"
    proof -
      from ⟨dip ∈ vD(rt (σ nhip))⟩ and ⟨dip ∈ vD(rt (σ nhip'))⟩
        have "dip ∈ vD(rt (σ nhip)) ∩ vD(rt (σ nhip'))" ..
      with ⟨nhip' ≠ dip⟩
        and ⟨nhip' = the (nhop (rt (σ nhip)) dip)⟩
        and inv
        show "rt (σ nhip) ⊑dip rt (σ nhip')"
        by (clarsimp simp: Let_def)
      qed
    finally show "rt (σ ip) ⊑dip rt (σ nhip')" .
  qed } note fresher = this

show "irrefl ((rt_graph σ dip) +)"
unfolding irrefl_def proof (intro allI notI)
  fix ip
  assume "(ip, ip) ∈ (rt_graph σ dip) +"
  moreover then have "dip ∈ vD(rt (σ ip))"
    and "ip ≠ dip"

  by auto
  ultimately have "rt (σ ip) ⊑dip rt (σ ip)" by (rule fresher)
  thus False by simp
qed
qed
end

```

## 5.11 Lift and transfer invariants to show loop freedom

theory E\_Aodv\_Loop\_Freedom

imports "../.../AWN/OClosed\_Transfer" "../.../AWN/Qmsg\_Lifting" E\_Global\_Invariants E\_Loop\_Freedom  
begin

### 5.11.1 Lift to parallel processes with queues

lemma par\_step\_no\_change\_on\_send\_or\_receive:

fixes  $\sigma$   $s$   $a$   $\sigma'$   $s'$   
assumes " $((\sigma, s), a, (\sigma', s')) \in \text{oparp\_sos } i \text{ (oseqp\_sos } \Gamma_{AODV} i) \text{ (seqp\_sos } \Gamma_{QMSG})$ "  
and " $a \neq \tau$ "  
shows " $\sigma' i = \sigma i$ "  
using assms by (rule qmsg\_no\_change\_on\_send\_or\_receive)

lemma par\_nhop\_quality\_increases:

shows " $\text{opaodv } i \llbracket_i \text{ qmsg} \models (\text{otherwith } (op=) \{i\} (\text{orecvmsg } (\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)), \text{other quality\_increases } \{i\} \rightarrow) \text{ global } (\lambda\sigma. \forall \text{dip. let nhop} = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip}) \text{ in dip} \in \text{vD } (\text{rt } (\sigma i)) \cap \text{vD } (\text{rt } (\sigma \text{nhop})) \wedge \text{nhop} \neq \text{dip} \rightarrow (\text{rt } (\sigma i)) \sqsubseteq_{\text{dip}} (\text{rt } (\sigma \text{nhop})))$ "

proof (rule lift\_into\_qmsg [OF seq\_nhop\_quality\_increases])

show " $\text{opaodv } i \models_A (\text{otherwith } (op=) \{i\} (\text{orecvmsg } (\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)), \text{other quality\_increases } \{i\} \rightarrow) \text{ globala } (\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

proof (rule ostep\_invariant\_weakenE [OF oquality\_increases], simp\_all)

fix  $t$  :: " $((\text{nat} \Rightarrow \text{state}) \times (\text{state}, \text{msg}, \text{pseqp}, \text{pseqp label}) \text{seqp}), \text{msg seq\_action}) \text{ transition}$ "

assume " $\text{onll } \Gamma_{AODV} (\lambda((\sigma, \_), \_, (\sigma', \_)). \forall j. \text{quality\_increases } (\sigma j) (\sigma' j)) t$ "

thus " $\text{quality\_increases } (\text{fst } (\text{fst } t) i) (\text{fst } (\text{snd } (\text{snd } t)) i)$ "

by (cases  $t$ ) (clarsimp dest!: onllD, metis aodv\_ex\_label)

next

fix  $\sigma$   $\sigma'$   $a$

assume " $\text{otherwith } (op=) \{i\} (\text{orecvmsg } (\lambda\sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)) \sigma \sigma' a$ "

thus " $\text{otherwith quality\_increases } \{i\} (\text{orecvmsg } (\lambda_. \text{rreq\_rrep\_sn})) \sigma \sigma' a$ "

by - (erule weaken\_otherwith, auto)

qed

qed auto

lemma par\_rreq\_rrep\_sn\_quality\_increases:

" $\text{opaodv } i \llbracket_i \text{ qmsg} \models_A (\lambda\sigma \_. \text{orecvmsg } (\lambda_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_. \text{True}) \{i\} \rightarrow) \text{ globala } (\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

proof -

have " $\text{opaodv } i \models_A (\lambda\sigma \_. \text{orecvmsg } (\lambda_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_. \text{True}) \{i\} \rightarrow) \text{ globala } (\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

globala  $(\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

by (rule ostep\_invariant\_weakenE [OF olocal\_quality\_increases])

(auto dest!: onllD seqllD elim!: aodv\_ex\_labelE)

hence " $\text{opaodv } i \llbracket_i \text{ qmsg} \models_A (\lambda\sigma \_. \text{orecvmsg } (\lambda_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_. \text{True}) \{i\} \rightarrow) \text{ globala } (\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

globala  $(\lambda(\sigma, \_, \sigma'). \text{quality\_increases } (\sigma i) (\sigma' i))$ "

by (rule lift\_step\_into\_qmsg\_statelessassm) simp\_all

thus ?thesis by rule auto

qed

lemma par\_rreq\_rrep\_nsqn\_fresh\_any\_step:

shows " $\text{opaodv } i \llbracket_i \text{ qmsg} \models_A (\lambda\sigma \_. \text{orecvmsg } (\lambda_. \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda\_ \_. \text{True}) \{i\} \rightarrow) \text{ globala } (\lambda(\sigma, a, \sigma'). \text{anycast } (\text{msg\_fresh } \sigma) a)$ "

proof -

have " $\text{opaodv } i \models_A (\lambda\sigma \_. (\text{orecvmsg } (\lambda_. \text{rreq\_rrep\_sn})) \sigma, \text{other } (\lambda\_ \_. \text{True}) \{i\} \rightarrow) \text{ globala } (\lambda(\sigma, a, \sigma'). \text{anycast } (\text{msg\_fresh } \sigma) a)$ "

globala  $(\lambda(\sigma, a, \sigma'). \text{anycast } (\text{msg\_fresh } \sigma) a)$ "

proof (rule ostep\_invariant\_weakenE [OF rreq\_rrep\_nsqn\_fresh\_any\_step\_invariant])

fix  $t$

```

assume "onll  $\Gamma_{AODV}$  ( $\lambda((\sigma, \_), a, \_)$ . anycast (msg_fresh  $\sigma$ ) a) t"
thus "globala ( $\lambda(\sigma, a, \sigma')$ . anycast (msg_fresh  $\sigma$ ) a) t"
  by (cases t) (clarsimp dest!: onllD, metis aadv_ex_label)
qed auto
hence "opaadv i  $\langle\langle_i$  qmsg  $\models_A$  ( $\lambda\sigma \_.$  orecvmsg ( $\lambda\_.$  rreq_rrep_sn))  $\sigma$ , other ( $\lambda\_.$  True)  $\{i\}$   $\rightarrow$ 
  globala ( $\lambda(\sigma, a, \sigma')$ . anycast (msg_fresh  $\sigma$ ) a)"
  by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

```

lemma par\_anycast\_msg\_zhops:

```

shows "opaadv i  $\langle\langle_i$  qmsg  $\models_A$  ( $\lambda\sigma \_.$  orecvmsg ( $\lambda\_.$  rreq_rrep_sn)  $\sigma$ , other ( $\lambda\_.$  True)  $\{i\}$   $\rightarrow$ 
  globala ( $\lambda(\_, a, \_)$ . anycast msg_zhops a)"

```

proof -

```

from anycast_msg_zhops initiali_aadv oaadv_trans aadv_trans

```

```

have "opaadv i  $\models_A$  (act TT, other ( $\lambda\_.$  True)  $\{i\}$   $\rightarrow$ 

```

```

  seqll i (onll  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_)$ . anycast msg_zhops a))"

```

```

  by (rule open_seq_step_invariant)

```

```

hence "opaadv i  $\models_A$  ( $\lambda\sigma \_.$  orecvmsg ( $\lambda\_.$  rreq_rrep_sn)  $\sigma$ , other ( $\lambda\_.$  True)  $\{i\}$   $\rightarrow$ 
  globala ( $\lambda(\_, a, \_)$ . anycast msg_zhops a)"

```

proof (rule ostep\_invariant\_weakenE)

```

fix t :: "((nat  $\Rightarrow$  state)  $\times$  (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"

```

```

assume "seqll i (onll  $\Gamma_{AODV}$  ( $\lambda(\_, a, \_)$ . anycast msg_zhops a)) t"

```

```

thus "globala ( $\lambda(\_, a, \_)$ . anycast msg_zhops a) t"

```

```

  by (cases t) (clarsimp dest!: seqllD onllD, metis aadv_ex_label)

```

qed simp\_all

```

hence "opaadv i  $\langle\langle_i$  qmsg  $\models_A$  ( $\lambda\sigma \_.$  orecvmsg ( $\lambda\_.$  rreq_rrep_sn)  $\sigma$ , other ( $\lambda\_.$  True)  $\{i\}$   $\rightarrow$ 
  globala ( $\lambda(\_, a, \_)$ . anycast msg_zhops a)"

```

```

  by (rule lift_step_into_qmsg_statelessassm) simp_all

```

```

thus ?thesis by rule auto

```

qed

## 5.11.2 Lift to nodes

lemma node\_step\_no\_change\_on\_send\_or\_receive:

```

assumes "(( $\sigma$ , NodeS i P R), a, ( $\sigma'$ , NodeS i' P' R'))  $\in$  onode_sos

```

```

  (oparp_sos i (oseqp_sos  $\Gamma_{AODV}$  i) (seqp_sos  $\Gamma_{QMSG}$ ))"

```

```

  and "a  $\neq$   $\tau$ "

```

```

  shows " $\sigma' i = \sigma i$ "

```

using assms

```

by (cases a) (auto elim!: par_step_no_change_on_send_or_receive)

```

lemma node\_nhop\_quality\_increases:

```

shows "< i : opaadv i  $\langle\langle_i$  qmsg : R  $\rangle_o$   $\models$ 
  (otherwith (op=)  $\{i\}$ 

```

```

  (oarrivmsg ( $\lambda\sigma m.$  msg_fresh  $\sigma m \wedge$  msg_zhops m)),

```

```

  other quality_increases  $\{i\}$ 

```

```

 $\rightarrow$  global ( $\lambda\sigma.$   $\forall$ dip. let nhop = the (nhop (rt ( $\sigma i$ )) dip)

```

```

  in dip  $\in$  vD (rt ( $\sigma i$ ))  $\cap$  vD (rt ( $\sigma$  nhop))  $\wedge$  nhop  $\neq$  dip

```

```

 $\rightarrow$  (rt ( $\sigma i$ ))  $\sqsubset_{dip}$  (rt ( $\sigma$  nhop)))"

```

```

by (rule node_lift [OF par_nhop_quality_increases]) auto

```

lemma node\_quality\_increases:

```

"< i : opaadv i  $\langle\langle_i$  qmsg : R  $\rangle_o$   $\models_A$  ( $\lambda\sigma \_.$  oarrivmsg ( $\lambda\_.$  rreq_rrep_sn)  $\sigma$ ,
  other ( $\lambda\_.$  True)  $\{i\}$   $\rightarrow$ 

```

```

  globala ( $\lambda(\sigma, \_, \sigma')$ . quality_increases ( $\sigma i$ ) ( $\sigma' i$ ))"

```

```

by (rule node_lift_step_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp

```

lemma node\_rreq\_rrep\_nsqn\_fresh\_any\_step:

```

shows "< i : opaadv i  $\langle\langle_i$  qmsg : R  $\rangle_o$   $\models_A$ 

```

```

  ( $\lambda\sigma \_.$  oarrivmsg ( $\lambda\_.$  rreq_rrep_sn)  $\sigma$ , other ( $\lambda\_.$  True)  $\{i\}$   $\rightarrow$ 

```

```

  globala ( $\lambda(\sigma, a, \sigma')$ . castmsg (msg_fresh  $\sigma$ ) a)"

```

```

by (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_nsqn_fresh_any_step])

```

lemma node\_anycast\_msg\_zhops:

shows " $\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R \rangle_o \models_A$   
 $(\lambda \sigma \_ . \text{oarrivmsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma, \text{other } (\lambda \_ \_ . \text{True}) \{i\} \rightarrow)$   
 $\text{globala } (\lambda (\_, a, \_). \text{castmsg } \text{msg\_zhops } a)$ "  
 by (rule node\_lift\_anycast\_statelessassm [OF par\_anycast\_msg\_zhops])

lemma node\_silent\_change\_only:

shows " $\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R_i \rangle_o \models_A$   $(\lambda \sigma \_ . \text{oarrivmsg } (\lambda \_ \_ . \text{True}) \sigma,$   
 $\text{other } (\lambda \_ \_ . \text{True}) \{i\} \rightarrow)$   
 $\text{globala } (\lambda (\sigma, a, \sigma'). a \neq \tau \rightarrow \sigma' i = \sigma i)$ "

proof (rule ostep\_invariantI, simp (no\_asm), rule impI)

fix  $\sigma \zeta a \sigma' \zeta'$

assume or: " $(\sigma, \zeta) \in \text{oreachable } (\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R_i \rangle_o$   
 $(\lambda \sigma \_ . \text{oarrivmsg } (\lambda \_ \_ . \text{True}) \sigma)$   
 $(\text{other } (\lambda \_ \_ . \text{True}) \{i\}))$ "

and tr: " $((\sigma, \zeta), a, (\sigma', \zeta')) \in \text{trans } (\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R_i \rangle_o)$ "

and " $a \neq \tau_n$ "

from or obtain p R where " $\zeta = \text{NodeS } i \text{ p } R$ "

by - (drule node\_net\_state, metis)

with tr have " $((\sigma, \text{NodeS } i \text{ p } R), a, (\sigma', \zeta'))$

$\in \text{onode\_sos } (\text{oparp\_sos } i (\text{trans } (\text{opaadv } i)) (\text{trans } \text{qmsg}))$ "

by simp

thus " $\sigma' i = \sigma i$ " using  $\langle a \neq \tau_n \rangle$

by (cases rule: onode\_sos.cases)

(auto elim: qmsg\_no\_change\_on\_send\_or\_receive)

qed

### 5.11.3 Lift to partial networks

lemma arrive\_rreq\_rrep\_nsqn\_fresh\_inc\_sn [simp]:

assumes " $\text{oarrivmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge P \sigma m) \sigma m$ "

shows " $\text{oarrivmsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma m$ "

using assms by (cases m) auto

lemma opnet\_nhop\_quality\_increases:

shows " $\text{opnet } (\lambda i. \text{opaadv } i \langle \langle_i \text{ qmsg} \rangle p \models$

$(\text{otherwith } (\text{op}=) (\text{net\_tree\_ips } p)$

$(\text{oarrivmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m)),$

$\text{other quality\_increases } (\text{net\_tree\_ips } p) \rightarrow)$

$\text{global } (\lambda \sigma. \forall i \in \text{net\_tree\_ips } p. \forall \text{dip}.$

$\text{let } \text{nhip} = \text{the } (\text{nhop } (\text{rt } (\sigma i)) \text{ dip})$

$\text{in } \text{dip} \in \text{vD } (\text{rt } (\sigma i)) \cap \text{vD } (\text{rt } (\sigma \text{nhip})) \wedge \text{nhip} \neq \text{dip}$

$\rightarrow (\text{rt } (\sigma i)) \sqsubseteq_{\text{dip}} (\text{rt } (\sigma \text{nhip}))$ )"

proof (rule pnet\_lift [OF node\_nhop\_quality\_increases])

fix i R

have " $\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R \rangle_o \models_A$   $(\lambda \sigma \_ . \text{oarrivmsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma,$   
 $\text{other } (\lambda \_ \_ . \text{True}) \{i\} \rightarrow) \text{globala } (\lambda (\sigma, a, \sigma').$

$\text{castmsg } (\lambda m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) a)$ "

proof (rule ostep\_invariantI, simp (no\_asm))

fix  $\sigma s a \sigma' s'$

assume or: " $(\sigma, s) \in \text{oreachable } (\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R \rangle_o$   
 $(\lambda \sigma \_ . \text{oarrivmsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma)$   
 $(\text{other } (\lambda \_ \_ . \text{True}) \{i\}))$ "

and tr: " $((\sigma, s), a, (\sigma', s')) \in \text{trans } (\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R \rangle_o)$ "

and am: " $\text{oarrivmsg } (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma a$ "

from or tr am have " $\text{castmsg } (\text{msg\_fresh } \sigma) a$ "

by (auto dest!: ostep\_invariantD [OF node\_rreq\_rrep\_nsqn\_fresh\_any\_step])

moreover from or tr am have " $\text{castmsg } (\text{msg\_zhops}) a$ "

by (auto dest!: ostep\_invariantD [OF node\_anycast\_msg\_zhops])

ultimately show " $\text{castmsg } (\lambda m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) a$ "

by (case\_tac a) auto

qed

thus " $\langle i : \text{opaadv } i \langle \langle_i \text{ qmsg} : R \rangle_o \models_A$

$(\lambda \sigma \_ . \text{oarrivmsg } (\lambda \sigma m. \text{msg\_fresh } \sigma m \wedge \text{msg\_zhops } m) \sigma,$

```

      other quality_increases {i} → globala (λ(σ, a, _).
        castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
    by rule auto
  next
  fix i R
  show "⟨i : opaadv i ⟨⟨i qmsg : R⟩_o ⟩_A
    (λσ _. oarrivmsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
      other quality_increases {i} →) globala (λ(σ, a, σ').
        a ≠ τ ∧ (∀i d. a = i:deliver(d)) → σ i = σ' i)"
    by (rule ostep_invariant_weakenE [OF node_silent_change_only]) auto
  next
  fix i R
  show "⟨i : opaadv i ⟨⟨i qmsg : R⟩_o ⟩_A
    (λσ _. oarrivmsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
      other quality_increases {i} →) globala (λ(σ, a, σ').
        a = τ ∨ (∃d. a = i:deliver(d)) → quality_increases (σ i) (σ' i))"
    by (rule ostep_invariant_weakenE [OF node_quality_increases]) auto
  qed simp_all

```

#### 5.11.4 Lift to closed networks

lemma onet\_nhop\_quality\_increases:

```

shows "oclosed (opnet (λi. opaadv i ⟨⟨i qmsg⟩ p)
  = (λ_ _ . True, other quality_increases (net_tree_ips p) →)
    global (λσ. ∀i∈net_tree_ips p. ∀dip.
      let nhop = the (nhop (rt (σ i)) dip)
      in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhop)) ∧ nhop ≠ dip
      → (rt (σ i)) ⊆dip (rt (σ nhop)))"

```

(is "\_ = ( \_, ?U →) ?inv")

proof (rule inclosed\_closed)

from opnet\_nhop\_quality\_increases

```

  show "opnet (λi. opaadv i ⟨⟨i qmsg⟩ p)
    = (otherwith (op=) (net_tree_ips p) inclosed, ?U →) ?inv"

```

proof (rule oinvariant\_weakenE)

fix σ σ' :: "ip ⇒ state" and a :: "msg node\_action"

assume "otherwith (op=) (net\_tree\_ips p) inclosed σ σ' a"

thus "otherwith (op=) (net\_tree\_ips p)

(oarrivmsg (λσ m. msg\_fresh σ m ∧ msg\_zhops m)) σ σ' a"

proof (rule otherwithEI)

fix σ :: "ip ⇒ state" and a :: "msg node\_action"

assume "inclosed σ a"

thus "oarrivmsg (λσ m. msg\_fresh σ m ∧ msg\_zhops m) σ a"

proof (cases a)

fix ii ni ms

assume "a = ii-ni:arrive(ms)"

moreover with (inclosed σ a) obtain d di where "ms = newpkt(d, di)"

by (cases ms) auto

ultimately show ?thesis by simp

qed simp\_all

qed

qed

qed

#### 5.11.5 Transfer into the standard model

interpretation aadv\_openproc: openproc paadv opaadv id

where "aadv\_openproc.initmissing = initmissing"

proof -

show "openproc paadv opaadv id"

proof unfold\_locales

fix i :: ip

have "{(σ, ζ). (σ i, ζ) ∈ σ<sub>AADV</sub> i ∧ (∀j. j ≠ i → σ j ∈ fst ' σ<sub>AADV</sub> j)} ⊆ σ<sub>AADV</sub>'"

unfolding σ<sub>AADV</sub>\_def σ<sub>AADV</sub>'\_def

proof (rule equalityD1)

```

    show "∧f p. {(σ, ζ). (σ i, ζ) ∈ {(f i, p)} ∧ (∀j. j ≠ i
      → σ j ∈ fst ' {(f j, p)}')} = {(f, p)}"
      by (rule set_eqI) auto
  qed
  thus "{ (σ, ζ) | σ ζ s. s ∈ init (paadv i)
    ∧ (σ i, ζ) = id s
    ∧ (∀j. j≠i → σ j ∈ (fst o id) ' init (paadv j)) } ⊆ init (opaadv i)"
    by simp
next
  show "∀j. init (paadv j) ≠ {}"
    unfolding σ_AODV_def by simp
next
  fix i s a s' σ σ'
  assume "σ i = fst (id s)"
    and "σ' i = fst (id s)"
    and "(s, a, s') ∈ trans (paadv i)"
  then obtain q q' where "s = (σ i, q)"
    and "s' = (σ' i, q)"
    and "((σ i, q), a, (σ' i, q')) ∈ trans (paadv i)"
    by (cases s, cases s') auto
  from this(3) have "((σ, q), a, (σ', q')) ∈ trans (opaadv i)"
    by simp (rule open_seqp_action [OF aadv_wf])

  with ⟨s = (σ i, q)⟩ and ⟨s' = (σ' i, q')⟩
  show "((σ, snd (id s)), a, (σ', snd (id s'))) ∈ trans (opaadv i)"
    by simp
  qed
  then interpret op: openproc paadv opaadv id .
  have [simp]: "∧i. (SOME x. x ∈ (fst o id) ' init (paadv i)) = aadv_init i"
    unfolding σ_AODV_def by simp
  hence "∧i. openproc.initmissing paadv id i = initmissing i"
    unfolding op.initmissing_def op.someinit_def initmissing_def
    by (auto split: option.split)
  thus "openproc.initmissing paadv id = initmissing" ..
qed

interpretation aadv_openproc_par_qmsg: openproc_parq paadv opaadv id qmsg
  where "aadv_openproc_par_qmsg.netglobal = netglobal"
    and "aadv_openproc_par_qmsg.initmissing = initmissing"
proof -
  show "openproc_parq paadv opaadv id qmsg"
    by (unfold_locales) simp
  then interpret opq: openproc_parq paadv opaadv id qmsg .

  have im: "∧σ. openproc.initmissing (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) σ
    = initmissing σ"
    unfolding opq.initmissing_def opq.someinit_def initmissing_def
    unfolding σ_AODV_def σ_QMSG_def by (clarsimp cong: option.case_cong)
  thus "openproc.initmissing (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) = initmissing"
    by (rule ext)
  have "∧P σ. openproc.netglobal (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) P σ
    = netglobal P σ"
    unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def
    unfolding σ_AODV_def σ_QMSG_def
    by (clarsimp cong: option.case_cong
      simp del: One_nat_def
      simp add: fst_initmissing_netgmap_default_aadv_init_netlift
        [symmetric, unfolded initmissing_def])
  thus "openproc.netglobal (λi. paadv i ⟨⟨ qmsg ⟩⟩ (λ(p, q). (fst (id p), snd (id p), q)) = netglobal"
    by auto
  qed

lemma net_nhop_quality_increases:
  assumes "wf_net_tree n"

```

```

shows "closed (pnet (λi. paadv i ⟨⟨ qmsg⟩ n) ⊨ netglobal
      (λσ. ∀i dip. let nhip = the (nhop (rt (σ i)) dip)
                  in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
                  → (rt (σ i)) ⊑dip (rt (σ nhip)))"
(is "_ ⊨ netglobal (λσ. ∀i. ?inv σ i)")
proof -
  from (wf_net_tree n)
  have proto: "closed (pnet (λi. paadv i ⟨⟨ qmsg⟩ n) ⊨ netglobal (λσ. ∀i∈net_tree_ips n. ∀dip.
    let nhip = the (nhop (rt (σ i)) dip)
    in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
    → (rt (σ i)) ⊑dip (rt (σ nhip)))"
  by (rule aadv_openproc_par_qmsg.close_opnet [OF _ onet_nhop_quality_increases])
show ?thesis
unfolding invariant_def opnet_sos.opnet_tau1
proof (rule, simp only: aadv_openproc_par_qmsg.netglobalsimp
  fst_initmissing_netgmap_pair_fst, rule allI)
  fix σ i
  assume sr: "σ ∈ reachable (closed (pnet (λi. paadv i ⟨⟨ qmsg⟩ n) TT"
  hence "∀i∈net_tree_ips n. ?inv (fst (initmissing (netgmap fst σ))) i"
  by - (drule invariantD [OF proto],
    simp only: aadv_openproc_par_qmsg.netglobalsimp
    fst_initmissing_netgmap_pair_fst)
  thus "?inv (fst (initmissing (netgmap fst σ))) i"
  proof (cases "i∈net_tree_ips n")
    assume "i∉net_tree_ips n"
    from sr have "σ ∈ reachable (pnet (λi. paadv i ⟨⟨ qmsg⟩ n) TT" ..
    hence "net_ips σ = net_tree_ips n" ..
    with (i∉net_tree_ips n) have "i∉net_ips σ" by simp
    hence "(fst (initmissing (netgmap fst σ))) i = aadv_init i"
    by simp
    thus ?thesis by simp
  qed metis
qed
qed

```

### 5.11.6 Loop freedom of AODV

```

theorem aadv_loop_freedom:
  assumes "wf_net_tree n"
  shows "closed (pnet (λi. paadv i ⟨⟨ qmsg⟩ n) ⊨ netglobal (λσ. ∀dip. irrefl ((rt_graph σ dip)+))"
  using assms by (rule aadv_openproc_par_qmsg.netglobal_weakenE
    [OF net_nhop_quality_increases inv_to_loop_freedom])
end

```



# Bibliography

- [1] T. Bourke. Mechanization of the Algebra for Wireless Networks (AWN). *Archive of Formal Proofs*, 2014. <http://afp.sf.net/entries/AWN.shtml>.
- [2] T. Bourke, R. J. van Glabbeek, and P. Höfner. A mechanized proof of loop freedom of the (untimed) AODV routing protocol. In F. Cassez and J.-F. Raskin, editors, *Proceedings of the 12th International Conference on Automated Technology for Verification and Analysis (ATVA 2014)*, volume 8837 of *Lecture Notes in Computer Science*, pages 47–63, Sydney, Australia, Nov. 2014. Springer.
- [3] T. Bourke, R. J. van Glabbeek, and P. Höfner. Showing invariance compositionally for a process algebra for network protocols. In G. Klein and R. Gamboa, editors, *Proceedings of the 5th International Conference on Interactive Theorem Proving (ITP 2014)*, volume 8558 of *Lecture Notes in Computer Science*, pages 144–159, Vienna, Austria, July 2014. Springer.
- [4] A. Fehnker, R. J. van Glabbeek, P. Höfner, A. McIver, M. Portmann, and W. L. Tan. A process algebra for wireless mesh networks used for modelling, verifying and analysing AODV. Technical Report 5513, NICTA, 2013.
- [5] S. Miskovic and E. W. Knightly. Routing primitives for wireless mesh networks: Design, analysis and experiments. In *Conference on Information Communications (INFOCOM '10)*, pages 2793–2801. IEEE, 2010.
- [6] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561 (Experimental), Network Working Group, 2003.