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-01104033
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\textsuperscript{1} \quad Peter Höfner\textsuperscript{2}

November 3, 2014

\textsuperscript{1}Inria, École normale supérieure, and NICTA
\textsuperscript{2}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show that Isabelle/HOL can re-establish most proof obligations automatically and identify exactly the steps that are no longer valid.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].
### 2 Variant B: Forwarding the Route Reply

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Predicates and functions used in the AODV model</td>
<td>170</td>
</tr>
<tr>
<td>2.1.1 Sequence Numbers</td>
<td>170</td>
</tr>
<tr>
<td>2.1.2 Modelling Routes</td>
<td>170</td>
</tr>
<tr>
<td>2.1.3 Routing Tables</td>
<td>171</td>
</tr>
<tr>
<td>2.1.4 Updating Routing Tables</td>
<td>174</td>
</tr>
<tr>
<td>2.1.5 Route Requests</td>
<td>184</td>
</tr>
<tr>
<td>2.1.6 Queued Packets</td>
<td>184</td>
</tr>
<tr>
<td>2.1.7 Comparison with the original technical report</td>
<td>185</td>
</tr>
<tr>
<td>2.2 AODV protocol messages</td>
<td>185</td>
</tr>
<tr>
<td>2.3 The AODV protocol</td>
<td>186</td>
</tr>
<tr>
<td>2.3.1 Data state</td>
<td>186</td>
</tr>
<tr>
<td>2.3.2 Auxilliary message handling definitions</td>
<td>188</td>
</tr>
<tr>
<td>2.3.3 The protocol process</td>
<td>189</td>
</tr>
<tr>
<td>2.4 Invariant assumptions and properties</td>
<td>195</td>
</tr>
<tr>
<td>2.5 Quality relations between routes</td>
<td>197</td>
</tr>
<tr>
<td>2.5.1 Net sequence numbers</td>
<td>197</td>
</tr>
<tr>
<td>2.5.2 Comparing routes</td>
<td>199</td>
</tr>
<tr>
<td>2.5.3 Comparing routing tables</td>
<td>200</td>
</tr>
<tr>
<td>2.5.4 Strictly comparing routing tables</td>
<td>205</td>
</tr>
<tr>
<td>2.6 Invariant proofs on individual processes</td>
<td>209</td>
</tr>
<tr>
<td>2.7 The quality increases predicate</td>
<td>219</td>
</tr>
<tr>
<td>2.8 The ‘open’ AODV model</td>
<td>226</td>
</tr>
<tr>
<td>2.9 Global invariant proofs over sequential processes</td>
<td>227</td>
</tr>
<tr>
<td>2.10 Routing graphs and loop freedom</td>
<td>245</td>
</tr>
<tr>
<td>2.11 Lift and transfer invariants to show loop freedom</td>
<td>247</td>
</tr>
<tr>
<td>2.11.1 Lift to parallel processes with queues</td>
<td>247</td>
</tr>
<tr>
<td>2.11.2 Lift to nodes</td>
<td>248</td>
</tr>
<tr>
<td>2.11.3 Lift to partial networks</td>
<td>249</td>
</tr>
<tr>
<td>2.11.4 Lift to closed networks</td>
<td>250</td>
</tr>
<tr>
<td>2.11.5 Transfer into the standard model</td>
<td>250</td>
</tr>
<tr>
<td>2.11.6 Loop freedom of AODV</td>
<td>252</td>
</tr>
</tbody>
</table>

### 3 Variant C: From Groupcast to Broadcast

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Predicates and functions used in the AODV model</td>
<td>253</td>
</tr>
<tr>
<td>3.1.1 Sequence Numbers</td>
<td>253</td>
</tr>
<tr>
<td>3.1.2 Modelling Routes</td>
<td>253</td>
</tr>
<tr>
<td>3.1.3 Routing Tables</td>
<td>254</td>
</tr>
<tr>
<td>3.1.4 Updating Routing Tables</td>
<td>257</td>
</tr>
</tbody>
</table>
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 Auxiliary 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

4 Variant D: Forwarding the Route Request .......................... 332

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 Auxiliary 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
5 Variants A–D: All proposed modifications

5.1 Predicates and functions used in the AODV model

5.1.1 Sequence Numbers

5.1.2 Modelling Routes

5.1.3 Routing Tables

5.1.4 Updating Routing Tables

5.1.5 Route Requests

5.1.6 Queued Packets

5.1.7 Comparison with the original technical report

5.2 AODV protocol messages

5.3 The AODV protocol

5.3.1 Data state

5.3.2 Auxiliary message handling definitions

5.3.3 The protocol process

5.4 Invariant assumptions and properties

5.5 Quality relations between routes

5.5.1 Net sequence numbers

5.5.2 Comparing routes

5.5.3 Comparing routing tables

5.5.4 Strictly comparing routing tables

5.6 Invariant proofs on individual processes

5.7 The quality increases predicate

5.8 The ‘open’ AODV model

5.9 Global invariant proofs over sequential processes

5.10 Routing graphs and loop freedom

5.11 Lift and transfer invariants to show loop freedom

5.11.1 Lift to parallel processes with queues

5.11.2 Lift to nodes

5.11.3 Lift to partial networks

5.11.4 Lift to closed networks

5.11.5 Transfer into the standard model

5.11.6 Loop freedom of AODV
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 ≡ Known"
abbreviation unk where "unk ≡ Unknown"
datatype p = NoRequestRequired | RequestRequired
abbreviation noreq where "noreq ≡ NoRequestRequired"
abbreviation req where "req ≡ RequestRequired"
datatype f = Valid | Invalid
abbreviation val where "val ≡ Valid"
abbreviation inv where "inv ≡ Invalid"

lemma not_ks [simp]:
"(x ≠ kno) = (x = unk)"
"(x ≠ unk) = (x = kno)"
by (cases x,clarsimp+)+

lemma not_ps [simp]:
"(x ≠ noreq) = (x = req)"
"(x ≠ req) = (x = noreq)"
by (cases x,clarsimp+)+

lemma not_ffs [simp]:
"(x ≠ val) = (x = inv)"
"(x ≠ 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 ⇒ sqn"
where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ 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 ≠ Suc 0"
unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 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 \(r = "sqn \times k \times f \times nat \times ip \times ip set"\)

definition proj2 :: "r \Rightarrow sqn" ("\pi_2")
  where "\pi_2 \equiv \lambda(dsn, _, _, _, _, _). dsn"

definition proj3 :: "r \Rightarrow k" ("\pi_3")
  where "\pi_3 \equiv \lambda(_, dsk, _, _, _, _). dsk"

definition proj4 :: "r \Rightarrow f" ("\pi_4")
  where "\pi_4 \equiv \lambda(_, _, flag, _, _, _). flag"

definition proj5 :: "r \Rightarrow nat" ("\pi_5")
  where "\pi_5 \equiv \lambda(_, _, _, hops, _, _). hops"

definition proj6 :: "r \Rightarrow ip" ("\pi_6")
  where "\pi_6 \equiv \lambda(_, _, _, _, nhip, _). nhip"

definition proj7 :: "r \Rightarrow ip set" ("\pi_7")
  where "\pi_7 \equiv \lambda(_, _, _, _, _, pre). pre"

lemma projs [simp]:
  "\pi_2(dsn, dsk, flag, hops, nhip, pre) = dsn"
  "\pi_3(dsn, dsk, flag, hops, nhip, pre) = dsk"
  "\pi_4(dsn, dsk, flag, hops, nhip, pre) = flag"
  "\pi_5(dsn, dsk, flag, hops, nhip, pre) = hops"
  "\pi_6(dsn, dsk, flag, hops, nhip, pre) = nhip"
  "\pi_7(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 ] \] \Rightarrow P (\pi_3 x)"
  by (rule k.induct)

lemma proj4_pred [intro]: "[\ [P val; P inv ] \] \Rightarrow P (\pi_4 x)"
  by (rule f.induct)

lemma proj6_pair_snd [simp]:
  fixes dsn' \(r\)
  shows "\pi_6 (dsn', snd (r)) = \pi_6(r)"
  by (cases r) simp

0.2.3 Routing Tables

Routing tables map ip addresses to route entries.

type_synonym \(rt = "ip \Rightarrow r"\)

syntax
  "\_Sigma_route" :: "rt \Rightarrow ip \Rightarrow r" ("\_\_\Sigma\_route(_, _)")

translations
  "\_\_\Sigma\_route(rt, dip)" \Rightarrow "rt dip"

definition sqn :: "rt \Rightarrow ip \Rightarrow sqn"
  where "sqn rt dip \equiv case \_\_\Sigma\_route(rt, 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\_route(rt, dip) of Some r \Rightarrow \pi_3(r) | None \Rightarrow unk"
abbreviation flag :: "rt ⇒ ip ⇒ f" where "flag rt dip ≡ map_option π₄ (σ_route(rt, dip))"

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

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

abbreviation precs :: "rt ⇒ ip ⇒ ip set" where "prec rt dip ≡ map_option π₇ (σ_route(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. σ_route(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 "σ_route(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. σ_route(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. σ_route(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 \implies \text{the (flag rt dip) = val}"
unfolding vD_def by clarsimp

lemma kD_update [simp]:
"∀ nip v. kD (rt nip \mapsto 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' \implies P rt ip ip'"
and "\[ ip \neq ip'; ip ∈ kD rt \]\implies P 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 ⇒ 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 "\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]:
"\forall 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]:
"\\forall rt dip ip npre. dip\in\ kD\ (rt) \Rightarrow \pi_5\ (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 \( \text{"nhop (the \ (\text{addpreRT \ rt \ dip \ pre)) \ ip} = \text{nhop \ rt \ ip"} \)

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 \( \text{"sqn \ (the \ (\text{addpreRT \ rt \ dip \ pre)) \ ip} = sqn \ rt \ ip"} \)

unfolding addpreRT_def
using assms [THEN kd_Some] by (clarsimp)

**lemma dhops_addpreRT [simp]:**

fixes rt pre ip dip
assumes "dip \in\ kd \ rt"
shows \( \text{"dhops \ (the \ (\text{addpreRT \ rt \ dip \ pre)) \ ip} = dhops \ rt \ ip"} \)

unfolding addpreRT_def
using assms [THEN kd_Some] by (clarsimp)

**lemma sqnf_addpreRT [simp]:**

\[ \forall ip \ dip. \ ip \in kD(rt) \Rightarrow sqnf \ (the \ (\text{addpreRT \ (rt \ dip \ pre)) \ ip} = sqnf \ (rt \ dip) \]

unfolding sqnf_def addpreRT_def by auto

**Updating route entries**

**lemma in_kD_case [simp]:**

fixes dip rt
assumes "dip \in\ kd \ rt"
shows \( \text{"(case \ rt \ dip \ of \ None} \Rightarrow \text{en} \mid \text{Some \ r} \Rightarrow \text{es \ r} \} = \text{es \ (the \ (rt \ dip))"} \)

using assms [THEN kd_Some] by auto

**lemma not_in_kD_case [simp]:**

fixes dip rt
assumes "dip \notin\ kd \ (rt)"
shows \( \text{"(case \ rt \ dip \ of \ None} \Rightarrow \text{en} \mid \text{Some \ r} \Rightarrow \text{es \ r} \} = \text{en"} \)

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 \( \text{"sqn \ rt \ ip} = \text{dsn"} \)

unfolding sqn_def using assms by simp

**lemma not_kD_sqn [simp]:**

fixes dip rt
assumes "dip \notin\ kd \ (rt)"
shows \( \text{"sqn \ rt \ dip} = 0" \)

using assms unfolding sqn_def by simp

**definition update_arg_wf :: "r \Rightarrow \text{bool}"**

where \( \text{"update_arg_wf \ r} = \pi_4(r) = \text{val} \land\)

\( (\pi_2(r) = 0) = (\pi_3(r) = \text{unk}) \land\)

\( (\pi_3(r) = \text{unk} \longrightarrow \pi_5(r) = 1)" \)

**lemma update_arg_wf_gives_cases:**

\( \forall r. \ update_arg_wf \ r \Rightarrow (\pi_2(r) = 0) = (\pi_3(r) = \text{unk})" \)

unfolding update_arg_wf_def by simp

**lemma update_arg_wf_tuples [simp]:**

"\forall \text{nhip \ pre. \ update_arg_wf \ (0, \ unk, \ val, \ Suc \ 0, \ \text{nhip}, \ \text{pre})}"

"\forall \text{n \ hops \ \text{nhip \ pre. \ update_arg_wf \ (Suc \ n, \ \text{kno}, \ val, \ hops, \ \text{nhip}, \ \text{pre})}"

unfolding update_arg_wf_def by auto

**lemma update_arg_wf_tuples' [elim]:**
"\(\forall n \text{ hops nhip pre. } 0 \leq n \implies \text{update}_{arg_{\text{wf}}} (n, \text{kno}, \text{val}, \text{hops}, \text{nhip}, \text{pre})\)"

unfolding \text{update}_{arg_{\text{wf}}}_{\text{def}} by \text{auto}

lemma \text{wf}_r_cases [intro]:
fixes \(P \ r\)
assumes "\text{update}_{arg_{\text{wf}}} r"
and \(c1: "\forall \text{nhip pre. } P (0, \text{unk, val, Suc 0, nhip, pre})"
and \(c2: "\forall \text{dsn hops nhip pre. } \text{dsn} > 0 \implies P (dsn, \text{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 \(\langle \text{update}_{arg_{\text{wf}}} r \rangle\)
have \(\text{wf1: } \text{flag} = \text{val}\)
and \(\text{wf2: } (dsn = 0) \implies (dsk = \text{unk})\)
and \(\text{wf3: } dsk = \text{unk} \implies (hops = 1)\)
unfolding \text{update}_{arg_{\text{wf}}}_{\text{def}} by \text{auto}

have "\(P (dsn, dsk, flag, hops, nhip, pre)\)"
proof (cases \(dsk\))
assume "\(dsk = \text{unk}\)"
moreover with \(\text{wf2} \ \text{wf3}\)
have "\(dsn = 0\)"
and "\(hops = \text{Suc 0}\)"
by \text{auto}
ultimately show ?thesis
using \(\langle \text{flag} = \text{val} \rangle\)
by \text{simp} (rule \(c1\))
next
assume "\(dsk = \text{kno}\)"
moreover with \(\text{wf2}\)
have "\(dsn > 0\)"
by \text{simp}
ultimately show ?thesis
using \(\langle \text{flag} = \text{val} \rangle\)
by \text{simp} (rule \(c2\))
qed
with \(\ast\) show "\(P \ r\)" by \text{simp}
qed

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

lemma \text{update_simps} [simp]:
fixes \(r \ s \ nr \ nr' \ ns \ rt \ ip\)
defines "\(s \equiv \text{the } \sigma_{\text{route}} (rt, ip)\)"
and "\(nr \equiv \text{addpre } r \ (\pi_7 (s))\)"
and "\(nr' \equiv (\pi_3 (s), \pi_3 (nr), \pi_4 (nr), \pi_5 (nr), \pi_6 (nr), \pi_7 (nr))\)"
and "\(ns \equiv \text{addpre } s \ (\pi_7 (r))\)"
shows
"[\(\text{ip} \not\in \text{kd}(rt)\)] \implies \text{update } rt \ ip \ r = rt (ip \mapsto r)"
"[\(\text{ip} \in \text{kd}(rt); \text{sqn } rt \ ip < \pi_2 (r)\)] \implies \text{update } rt \ ip \ r = rt (ip \mapsto nr)"
"[\(\text{ip} \in \text{kd}(rt); \text{sqn } rt \ ip = \pi_2 (r);\)"
\(\text{the } (\text{dhops } rt \ ip) > \pi_5 (r)\)] \implies \text{update } rt \ ip \ r = rt (ip \mapsto nr)"
"[\(\text{ip} \in \text{kd}(rt); \text{sqn } rt \ ip = \pi_2 (r);\)"
\(\text{flag } rt \ ip = \text{Some inv}\)] \implies \text{update } rt \ ip \ r = rt (ip \mapsto nr)"
"[\(\text{ip} \in \text{kd}(rt); \pi_3 (r) = \text{unk}; (\pi_2 (r) = 0) = (\pi_3 (r) = \text{unk})\)] \implies \text{update } rt \ ip \ r = rt (ip \mapsto nr')"
"[\(\text{ip} \in \text{kd}(rt); \text{sqn } rt \ ip \geq \pi_2 (r);\)"
\(\pi_3 (r) = \text{kno};\)"
\(\text{sqn } rt \ ip = \pi_2 (r) \implies \text{the } (\text{dhops } rt \ ip) \leq \pi_5 (r) \land \text{the } (\text{flag } rt \ ip) = \text{val}\) ]
\implies \text{update } rt \ ip \ r = rt (ip \mapsto ns)"

proof -
assume "\(\text{ip} \not\in \text{kd}(rt)\)"
hence "\(\sigma_{\text{route}} (rt, ip) = \text{None}\) ..
thus "\(\text{update } rt \ ip \ r = rt (ip \mapsto r)\)"
unfolding \text{update}_\text{def} by \text{simp}
assume "ip ∈ kD(rt)"
    and "sqn rt ip < π\(_2\)(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 < π\(_2\)(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 = π\(_2\)(r)"
    and "the (dhops rt ip) > π\(_5\)(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 = π\(_2\)(r)⟩ and ⟨the (dhops rt ip) > π\(_5\)(r)⟩
    show "update rt ip r = rt (ip ↦ nr)"
    unfolding update_def nr_def s_def by auto

next
assumes "(π\(_3\)(r) = 0) = (π\(_3\)(r) = unk)"
and otherassms: "sqn rt ip ≥ π\(_2\)(r)"
    "π\(_3\)(r) = kno"
    "sqn rt ip = π\(_2\)(r) ⟷ the (dhops rt ip) ≤ π\(_5\)(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 ⟨(π\(_2\)(r) = 0) = (π\(_3\)(r) = unk)⟩ and ⟨(π\(_3\)(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 ≥ π\(_2\)(r)"
    "π\(_3\)(r) = kno"
    "sqn rt ip = π\(_2\)(r) ⟷ the (dhops rt ip) ≤ π\(_5\)(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 "(π\(_2\)(r) = 0) = (π\(_3\)(r) = unk)"
and c1: "[| ip /∈ kD(rt) |] ⟹ P (rt (ip ↦ r))"
and c2: "[| ip ∈ kD(rt); sqn rt ip < π\(_2\)(r) |]
    ⟹ P (rt (ip ↦ addpre r (π\(_7\)(the σroute(rt, ip))))))"
and c3: "[| ip ∈ kD(rt); sqn rt ip = π\(_2\)(r); the (dhops rt ip) > π\(_5\)(r) |]
    ⟹ P (rt (ip ↦ addpre r (π\(_7\)(the σroute(rt, ip))))))"
and c4: "[| ip ∈ kD(rt); sqn rt ip = π\(_2\)(r); the (flag rt ip) = inv |]
    ⟹ P (rt (ip ↦ addpre r (π\(_7\)(the σroute(rt, ip))))))"
and c5: "[| ip ∈ kD(rt); π\(_3\)(r) = unk |]
    ⟹ P (rt (ip ↦ (π\(_2\)(the σroute(rt, ip)), π\(_3\)(r),
        π\(_4\)(r), π\(_5\)(r), π\(_6\)(r), π\(_7\)(addpre r (π\(_7\)(the σroute(rt, ip)))))))))"
and c6: "[\{ ip \in kD(rt); sqn rt ip \geq \pi_2(r); \pi_3(r) = kno; sqn rt ip = \pi_2(r) \implies \text{(hops rt ip) \leq \pi_5(r) \land \text{(flag rt ip) = val} \}
\implies \text{P (rt (ip \mapsto \text{addpre (the } \sigma_{route}(rt, 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 \langle \pi_2(r) = 0 \rangle = \langle \pi_3(r) = unk \rangle\n
c2 [OF \langle \pi_3(r) = unk \rangle]

c3 [OF \langle \pi_3(r) = unk \rangle]

c4 [OF \langle \pi_3(r) = unk \rangle]

c5 [OF \langle \pi_3(r) = unk \rangle]

c6 [OF \langle \pi_3(r) = unk \rangle]

unfolding update_def sqn_def by auto

qed

lemma update_cases_kD:

assumes "(\pi_2(r) = 0) = (\pi_3(r) = unk)"

and \"ip \in kD(rt)\"

and c2: "sqn rt ip < \pi_2(r) \implies \text{P (rt (ip \mapsto \text{addpre r (the } \pi_7(\text{the } \sigma_{route}(rt, ip)))))}"

and c3: "[sqn rt ip = \pi_2(r); \text{(hops rt ip) > \pi_5(r)]}
\implies \text{P (rt (ip \mapsto \text{addpre r (the } \pi_7(\text{the } \sigma_{route}(rt, ip)))))}"

and c4: "[sqn rt ip = \pi_2(r); \text{(flag rt ip) = inv]}\n\implies \text{P (rt (ip \mapsto \text{addpre r (the } \pi_7(\text{the } \sigma_{route}(rt, ip)))))}"

and c5: "\pi_3(r) = unk \implies \text{P (rt (ip \mapsto \pi_7(\text{the } \sigma_{route}(rt, ip)), \pi_3(r), \pi_4(r), \pi_5(r), \pi_6(r), \pi_7(\text{addpre r (the } \sigma_{route}(rt, ip)))))}"

and c6: "[sqn rt ip \geq \pi_2(r); \pi_3(r) = kno; sqn rt ip = \pi_2(r) \implies \text{(hops rt ip) \leq \pi_5(r) \land \text{(flag rt ip) = val} \}
\implies \text{P (rt (ip \mapsto \text{addpre (the } \sigma_{route}(rt, ip))) (\pi_7(r)))}]"

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 \text{addpre r (the } \pi_7(\text{the } (rt ip))))))" by (rule c2)

next

assume "sqn rt ip = \pi_2(r)"

and "\text{(hops rt ip) > \pi_5(r)}"

thus "P (rt(ip \mapsto \text{addpre r (the } \pi_7(\text{the } (rt ip))))))" by (rule c3)

next

assume "sqn rt ip = \pi_2(r)"

and "\text{(flag rt ip) = inv}"

thus "P (rt(ip \mapsto \text{addpre r (the } \pi_7(\text{the } (rt ip))))))" by (rule c4)

next

assume "\pi_3(r) = unk"

thus "P (rt (ip \mapsto \pi_7(\text{the } \sigma_{route}(rt, ip)), \pi_3(r), \pi_4(r), \pi_5(r), \pi_6(r), \pi_7(\text{addpre r (the } \sigma_{route}(rt, ip))))))" by (rule c5)

next

assume "sqn rt ip \geq \pi_2(r)"

and "\pi_3(r) = kno"

and "sqn rt ip = \pi_2(r) \implies \text{(hops rt ip) \leq \pi_5(r) \land \text{(flag rt ip) = val}"

thus "P (rt (ip \mapsto \text{addpre (the } (rt ip)) (\pi_7(r)))))"
by (rule c6)
qed (simp add: \( \{ i p \in kD(\text{rt}) \} \))

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: "\( \forall v \cdot rt \ dip = Some v \implies update rt dip (dsn, dsk, flag, hops, nhip, {\}) \neq rt(dip \mapsto \text{addpre (the (rt dip)) (\pi_7 (dsn, dsk, flag, hops, nhip, {\}))})\)"
    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 nip nhip pre
assumes ex: "\( \forall ip \in kD \text{rt} \cdot the (dhops rt ip) \geq 1 \)"
  and ip: "(ip = rip \wedge Suc 0 \leq hops) \lor (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∈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 = πₛ 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 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 flg hops sip.
rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})

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 sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

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]:
"\(dip \in vD(update \ rt \ dip' \ (dsn, \ dsk, \ val, \ hops, \ nhip, \ pre)) \implies (dip \in vD(rt) \lor dip=dip')\)"

Unfolding update_def vD_def by (clarsimp split: option.split_asm split_if_asm)

Invalidating route entries

Definition invalidate :: "\(rt \Rightarrow (ip \rightarrow sqn) \Rightarrow rt\)"
Where "invalidate rt dests \(\equiv \lambda ip. \ \text{case} \ (rt \ ip, \ dests \ ip) \ \text{of} \"
\(\text{(None, \ _)} \Rightarrow \text{None} \)
\(\text{| (Some s, \ None)} \Rightarrow \text{Some s} \)
\(\text{| (Some (\ _, \ dsk, \ _, \ hops, \ nhip, \ pre), \ Some rsn)} \Rightarrow \)
\(\text{Some (rsn, \ dsk, \ inv, \ hops, \ nhip, \ pre)}\)"

Lemma proj3_invalidate [simp]:
"\(\forall dip. \ \pi_3((\text{invalidate rt dests dip})) = \pi_3((\text{the (rt dip)}))\)"
Unfolding invalidate_def by (clarsimp split: option.split)

Lemma proj5_invalidate [simp]:
"\(\forall dip. \ \pi_5((\text{invalidate rt dests dip})) = \pi_5((\text{the (rt dip)}))\)"
Unfolding invalidate_def by (clarsimp split: option.split)

Lemma proj6_invalidate [simp]:
"\(\forall dip. \ \pi_6((\text{invalidate rt dests dip})) = \pi_6((\text{the (rt dip)}))\)"
Unfolding invalidate_def by (clarsimp split: option.split)

Lemma proj7_invalidate [simp]:
"\(\forall dip. \ \pi_7((\text{invalidate rt dests dip})) = \pi_7((\text{the (rt dip)}))\)"
Unfolding invalidate_def by (clarsimp split: option.split)

Lemma invalidate_kD_inv [simp]:
"\(\forall rt \ dests. \ kD((\text{invalidate rt dests}) \ rt) = kD \ rt\)"
Unfolding invalidate_def kD_def by (simp split: option.split)

Lemma invalidate_sqn:
Fixes \(rt \ dip \ dests\)
Assumes "\(\forall rsn. \ dests \ dip = \text{Some (rsn)} \rightarrow sqn rt \ dip \leq rsn\)"
Shows "\(sqn rt \ dip \leq sqn (\text{invalidate rt dests} \ dip)\)"
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 "\(rt \ dip = \text{Some (dsn, dsk, flag, hops, nhip, pre)}\)"
By (metis kD_Some)
With assms show "\(sqn rt \ dip \leq sqn (\text{invalidate rt dests} \ dip)\)"
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 "\(\text{dests ipa = Some rsn}\)"
And "\(ipa \in kD(rt)\)"
Shows "\(sqn (\text{invalidate rt dests}) \ ipa = \ rsn\)"
Unfolding invalidate_def sqn_def
Using assms(1) assms(2) [THEN kD_Some]
By clarsimp

Lemma dhops_invalidate [simp]:
"\(\forall dip. \ \text{the (dhops (invalidate rt dests) dip) = the (dhops rt dip)}\)"
Unfolding invalidate_def by (clarsimp split: option.split)

Lemma sqnf_invalidate [simp]:
"\(\forall dip. \ sqnf (\text{invalidate rt \ dip}) (\text{dests \ dip}) = \text{sqnf (rt \ dip) dip}\)"
Unfolding sqnf_def invalidate_def by (clarsimp split: option.split)
lemma nhop_invalidate [simp]:
"\( \forall \text{dip} \cdot \text{the} \ (\text{nhop} \ (\text{invalidate} \ (\text{rt} \ \zeta) \ (\text{dests} \ \zeta)) \ \text{dip}) = \text{the} \ (\text{nhop} \ (\text{rt} \ \zeta) \ \text{dip}) \)"
unfolding invalidate_def by (clarsimp split: option.split)

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

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

lemma \( vD_{\text{invalidate}} vD_{\text{not_dests}} \): 
"\( \forall \text{dip} \ \text{rt} \ \text{dests} \cdot \text{dip} \in vD(\text{invalidate} \ \text{rt} \ \text{dests}) \Rightarrow \text{dip} \in vD(\text{rt}) \land \text{dests dip} = \text{None} \)"
unfolding invalidate_def \( vD_{\text{def}} \)
by (clarsimp split: option.split_asm)

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

lemma invalidate_changes:
fixes rt dests dip dsn dsk flag hops nhip pre
assumes "\( \text{invalidate} \ \text{rt} \ \text{dests} \ \text{dip} = \text{Some} \ (\text{dsn}, \ \text{dsk}, \ \text{flag}, \ \text{hops}, \ \text{nhip}, \ \text{pre}) \)"
shows "\( \text{dsn} = (\text{case dests dip of None \Rightarrow } \pi_2(\text{the} \ (\text{rt dip})) \ | \ \text{Some } rsn \Rightarrow rsn) \land \text{dsk} = \pi_3(\text{the} \ (\text{rt dip})) \land \text{flag} = (\text{if dests dip = None then } \pi_4(\text{the} \ (\text{rt dip})) \ \text{else inv}) \land \text{hops} = \pi_5(\text{the} \ (\text{rt dip})) \land \text{nhip} = \pi_6(\text{the} \ (\text{rt dip})) \land \text{pre} = \pi_7(\text{the} \ (\text{rt dip})) \)"
using assms unfolding invalidate_def
by (cases "\text{rt dip} \ \text{dip} \ \text{dests dip}\) auto

lemma proj3_inv: "\( \forall \text{dip} \ \text{rt} \ \text{dests} \cdot \text{dip} \in kD(\text{rt}) \Rightarrow \pi_3(\text{the} \ (\text{invalidate} \ \text{rt} \ \text{dests} \ \text{dip}) = \pi_3(\text{the} \ (\text{rt dip})) \)"
by (clarsimp simp: invalidate_def kD_def split: option.split)

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

0.2.5 Route Requests

Generate a fresh route request identifier.

definition nrreqid :: "(ip × rreqid) set ⇒ ip ⇒ rreqid"
where "nrreqid rreqs ip ≡ Max \( \{\text{n. (ip, n)} \in rreqs \cup \{0\} \} + 1 \)"

0.2.6 Queued Packets

Functions for sending data packets.

type_synonym store = "ip ⇒ (p × data list)"
definition \texttt{sigma_queue :: "store \Rightarrow ip \Rightarrow data list" ("σ_{queue}(\_ \_")})
  where \texttt{"σ_{queue}(store, dip) \equiv case store dip of None \Rightarrow [] | Some (p, q) \Rightarrow q"}

definition \texttt{qD :: "store \Rightarrow ip set"}
  where \texttt{"qD \equiv dom"}

definition \texttt{add :: "data \Rightarrow ip \Rightarrow store \Rightarrow store"}
  where \texttt{"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 \texttt{qD_add [simp]}
  fixes d dip store
  shows \texttt{"qD(add d dip store) = insert dip (qD store)"}
  unfolding \texttt{add_def Let_def qD_def}
  by (clarsimp split: option.split)

definition \texttt{drop :: "ip \Rightarrow store \mapsto store"}
  where \texttt{"drop dip store \equiv map_option (λ (p, q). if tl q = [] then store (dip := None)
       else store (dip \mapsto (p, tl q))) (store dip)"}

definition \texttt{sigma_p_flag :: "store \Rightarrow ip \Rightarrow p" ("σ_{p-flag}(\_ \_")})
  where \texttt{"σ_{p-flag}(store, dip) \equiv map_option fst (store dip)"}

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

definition \texttt{setRRF :: "store \Rightarrow (ip \Rightarrow sqn) \Rightarrow store"}
  where \texttt{"setRRF store dests \equiv λ dip. if dests dip = None then store dip
       else map_option (λ(_, 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. \texttt{nhop} is partial, thus a ‘the’ is needed, similarly for \texttt{dhops} and \texttt{addpreRT}.
2. \texttt{precs} is partial.
3. \texttt{σ_{p-flag}(store, dip)} is partial.
4. The routing table (\texttt{rt}) is modelled as a map (\texttt{ip \Rightarrow r option}) rather than a set of 7-tuples, likewise, the \texttt{r}
   is a 6-tuple rather than a 7-tuple, i.e., the destination ip-address (\texttt{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 \texttt{invalidate}, and \texttt{store}.

end

0.3 AODV protocol messages
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

0.4 The AODV protocol

theory Aodv
imports Aodv_Data Aodv_Message
  "./AWN/AWN_SOS_Lables" "./AWN/AWN_Invariants"
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 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),
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)
lemma clear_locals_sip_not_ip [simp]: "¬ (sip (clearlocals ξ) = ip ξ)"
unfolding clear_locals_def by simp

lemma clear_locals_but_not Globals [simp]:
  "ip (clearlocals ξ) = ip ξ"
  "sn (clearlocals ξ) = sn ξ"
  "rt (clearlocals ξ) = rt ξ"
  "rreqs (clearlocals ξ) = rreqs ξ"
  "store (clearlocals ξ) = store ξ"
unfolding clearlocals_def by auto

0.4.2 Auxiliary 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' ⇒
  _ ⇒ {}"

lemma is_rrep_asm [dest!]:
  assumes "ξ' ∈ is_rrep ξ"
  shows "(∃hops' dip' dsn' oip' sip'.
       msg ξ = Rrep hops' dip' dsn' oip' 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+) +

0.4.3 The protocol process

datatype pseqp =
  PAodv
| PNewPkt
| PPkt
| PRreq
fun nat_of_seqp :: "pseqp ⇒ 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) = 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 Γ_AODV :: "(state, msg, pseqp, pseqp label) seqp_env"
where
  "Γ_AODV PAodv = labelled PAodv (
receive(λmsg'. ξ (msg := msg' )),
( λis_newpkt. NEWPkt(λξ. (data ξ, ip ξ)))
  ◦ (λis_pkt. Pkt(λξ. (data ξ, dip ξ, oip ξ)))
  ◦ (λis_rreq)
    [ξ, ξ (rt := update (rt ξ) (sip ξ) (o, unk, val, 1, sip ξ, {}))]
    RREQ(λξ. (hops ξ, rreqid ξ, dip ξ, dns ξ, oip ξ, osn ξ, sip ξ))
  ◦ (λis_rrep)
    [ξ, ξ (rt := update (rt ξ) (sip ξ) (o, unk, val, 1, sip ξ, {}))]
    RREP(λξ. (hops ξ, dip ξ, dns ξ, oip ξ, sip ξ))
  ◦ (λis_rerr)
    [ξ, ξ (rt := update (rt ξ) (sip ξ) (o, 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()

  ◦ (λξ. { ξ (dip := dip)) | dip. dip ∈ qD(store ξ) ∩ vD(rt ξ)})

    [ξ, ξ (store := unsetRRF (store ξ) (dip ξ))]  

    [ξ, ξ (sn := inc (sn ξ))]  

    [ξ, ξ (rreqid := nrrreqid (rreqs ξ) (ip ξ))]  

    [ξ, ξ (rreq := nrrreq (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. dip ξ ∈ vD (rt ξ))

    [ξ, ξ (store := update (rt ξ) (sip ξ) (o, unk, val, 1, sip ξ, {}))]

    RREQ(λξ. (hops ξ, rreqid ξ, dip ξ, dns ξ, oip ξ, osn ξ, sip ξ))

  ◦ (ξ. dip ξ ∈ iD (rt ξ))

    groupcast(λξ. the (nhop (rt ξ) (dip ξ)),
              λξ. rerr([dip ξ → sqn (rt ξ) (dip ξ), ip ξ]). AODV()")

  ◦ (ξ. dip ξ ∉ iD (rt ξ))
ΓAODV PRreq = labelled PRreq (⟨ξ. (oip ξ, rreqid ξ) ∈ rreqs ξ⟩ AODV())
① ⟨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⟩ 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 ξ dip 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 ξ 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())
② ⟨ξ. dip ξ /≠ ip ξ⟩
  (⟨ξ. dip ξ 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 ξ dip 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 ξ 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())
③ ⟨ξ. oip ξ ξ vD (rt ξ)⟩
  (⟨ξ. oip ξ 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 ξ dip ξ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ)) then Some (inc (sqn (rt ξ) rip)) else None⟩ {}]
Γ

aodv_proc_cases [dest]:

lemma where

Γ

declare

proof

fixes

| "shows |

| "ctermsl (λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()"

Π ⟨ξ. oip ξ ∈ vD (rt ξ)⟩

AODV()

)

)

Π ⟨ξ. rt ξ = update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {}) ⟩

AODV()

"

"Γ_AODV PRerr = labelled PRerr ( [κ. 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) ] [κ. ξ () store := setRFF (store ξ) (dests ξ) ] [κ. ξ () pre := (λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)).AODV()]

Π ⟨ξ. oip ξ ∈ vD (rt ξ)⟩

AODV()

)

Π ⟨ξ. rt ξ = update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {}) ⟩

AODV()

"

declare Γ_AODV.simps [simp del, code del]

lemmas Γ_AODV.simps [simp, code] = Γ_AODV.simps [simplified]

fun Γ_AODV_skeleton

where

"Γ_AODV_skeleton PAodv = seqp_skeleton (Γ_AODV PAodv)"

/ "Γ_AODV_skeleton PNewPkt = seqp_skeleton (Γ_AODV PNewPkt)"

/ "Γ_AODV_skeleton PPkt = seqp_skeleton (Γ_AODV PPkt)"

/ "Γ_AODV_skeleton PRreq = seqp_skeleton (Γ_AODV PRreq)"

/ "Γ_AODV_skeleton PRrep = seqp_skeleton (Γ_AODV PRrep)"

/ "Γ_AODV_skeleton PRerr = seqp_skeleton (Γ_AODV PRerr)"

lemma Γ_AODV_skeleton_wf [simp]:

"wellformed Γ_AODV_skeleton"

proof (rule, intro allI)

fix pn pn'

show "call(pn') ∉ ctermsl (Γ_AODV_skeleton pn)"

by (cases pn) simp_all

qed

declare Γ_AODV_skeleton.simps [simp del, code del]

lemmas Γ_AODV_skeleton.simps [simp, code]

= Γ_AODV_skeleton.simps [simplified Γ_AODV.simps seqp_skeleton.simps]

lemma aodv_proc_cases [dest]:

fixes p pn

shows "p ∈ ctermsl (Γ_AODV pn) ⟷

(p ∈ ctermsl (Γ_AODV PAodv) ∨
 p ∈ ctermsl (Γ_AODV PNewPkt) ∨
 p ∈ ctermsl (Γ_AODV PPkt) ∨
 p ∈ ctermsl (Γ_AODV PRreq) ∨
 p ∈ ctermsl (Γ_AODV PRrep) ∨
 p ∈ ctermsl (Γ_AODV PRerr))"

using assms

by (cases pn) simp_all

definition σ_AODV :: "ip ⇒ (state × (state, msg, pseqp, pseqp label) seqp) set"
where \( \sigma^{AODV} \ i \equiv \{(aodv\_init \ i, \ \Gamma^{AODV} \ P_{AODV})\} \)

abbreviation \(paodv\) :: 

\[
\text{ip} \Rightarrow (\text{state} \times (\text{state}, \text{msg}, \text{pseqp}, \text{pseqp label}) \text{seqp}, \text{msg seq_action}) \text{automaton}
\]

where

\(\text{paodv} \ i \equiv (\text{init} = \sigma^{AODV} \ i, \ \text{trans} = \text{seqp sos} \ \Gamma^{AODV})\)\n
lemma \(aodv\_trans\): \(\text{trans} (\text{paodv} \ i) = \text{seqp sos} \ \Gamma^{AODV}\)

by simp

lemma \(aodv\_control\_within\) \([\text{simp}]\): \(\text{control\_within} \ \Gamma^{AODV} \ (\text{init} (\text{paodv} \ i))\)

unfolding \(\sigma^{AODV}\_def\) by (rule control\_withinI) (auto simp del: \(\Gamma^{AODV}\_\text{simps}\))

lemma \(aodv\_wf\) \([\text{simp}]\):

\"wellformed \ \Gamma^{AODV}\"  
proof (rule, intro allI)

fix \(pn \ pn'\)

show \("\text{call}(pn') \notin \text{stermsl} \ (\Gamma^{AODV} \ pn)\"

by (cases \(pn\)) simp_all

qed

lemmas \(aodv\_labels\_not\_empty\) \([\text{simp}]\) = labels\_not\_empty \([\text{OF aodv}\_wf]\)

lemma \(aodv\_ex\_label\) \([\text{intro}]\): \(\exists l. l \in \text{labels} \ \Gamma^{AODV} \ p\)

by (metis aodv\_labels\_not\_empty all\_not\_in_conv)

lemma \(aodv\_ex\_labelE\) \([\text{elim}]\):

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 \(aodv\_simple\_labels\) \([\text{simp}]\): \"\text{simple_labels} \ \Gamma^{AODV}\"

proof

fix \(pn \ p\)

assume \("p \in \text{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 \(\sigma^{AODV}\_labels\) \([\text{simp}]\): \("(\xi, \ p) \in \sigma^{AODV} \ i \implies \text{labels} \ \Gamma^{AODV} \ p = \{P_{AODV}=-0\}\"

unfolding \(\sigma^{AODV}\_def\) by simp

lemma \(aodv\_init\_kD\_empty\) \([\text{simp}]\):

"\((\xi, \ p) \in \sigma^{AODV} \ i \implies \text{kD} \ (\text{rt} \ \xi) = \{\}\"

unfolding \(\sigma^{AODV}\_def\) kD\_def by simp

lemma \(aodv\_init\_sip\_not\_ip\) \([\text{simp}]\): \("\neg(\text{sip} \ (\text{aodv}\_init \ i) = i)\"\) by simp

lemma \(aodv\_init\_sip\_not\_ip'\) \([\text{simp}]\):

assumes \("(\xi, \ p) \in \sigma^{AODV} \ i\"

shows \("\text{sip} \ \xi \neq \text{ip} \ \xi\"

using assms unfolding \(\sigma^{AODV}\_def\) by simp

lemma \(aodv\_init\_sip\_not\_i\) \([\text{simp}]\):

assumes \("(\xi, \ p) \in \sigma^{AODV} \ i\"

shows \("\text{sip} \ \xi \neq i\"

using assms unfolding \(\sigma^{AODV}\_def\) by simp

lemma \(clear\_locals\_sip\_not\_ip'\):

assumes \("\text{ip} \ \xi = i\"

shows \("\neg(\text{sip} \ (\text{clear\_locals} \ \xi) = i)\"

using assms by auto

29
Stop the simplifier from descending into process terms.

```
declare seqp_congs [cong]
```

Configure the main invariant tactic for AODV.

```
declare
  Γ_AODV_simps [cterms_env]
  aodv_proc_cases [ctermsl_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
```

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

lemma msg_sender_simps [simp]:
"\(\forall hops rreqid dip dsn dsk oip osn sip. \\
  \text{msg_sender (Rreq hops rreqid dip dsn dsk oip osn sip)} = sip\)"
"\(\forall hops dip dsn oip sip. \text{msg_sender (Rrep hops dip dsn oip sip)} = sip\)"
"\(\forall dests sip. \text{msg_sender (Rerr dests sip)} = sip\)"
"\(\forall d dip. \text{msg_sender (Pkt d dip sip)} = sip\)"

unfolding msg_sender_def by simp_all

definition msg_zhops :: "msg ⇒ bool"
where "msg_zhops m ≡ case m of Rreq rhapsc _ _ _ _ _ _ osnc _ ⇒ osnc ≥ 1 \\
                      | Rrep _ _ _ _ dipc _ _ sipc ⇒ dipc = sipc
                      | Rerr _ _ _ _ _ _ _ _ => True"

lemma msg_zhops_simps [simp]:
"\(\forall hops rreqid dip dsn dsk oip osn sip. \\
  \text{msg_zhops (Rreq hops rreqid dip dsn dsk oip osn sip)} = \text{hops = 0 ⇒ oip = sip} \)"
"\(\forall hops dip dsn oip sip. \text{msg_zhops (Rrep hops dip dsn oip sip)} = \text{hops = 0 ⇒ dip = sip} \)"
"\(\forall dests sip. \text{msg_zhops (Rerr dests sip)} = True\)"
"\(\forall d dip. \text{msg_zhops (Newpkt d dip)} = True\)"
"\(\forall d dip. \text{msg_zhops (Pkt d dip sip)} = True\)"

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 _ _ dip _ _ _ _ _ _ _ ⇒ dsnc ≥ 1 \\
                      | _ ⇒ True"

lemma rreq_rrep_sn_simps [simp]:
"\(\forall hops rreqid dip dsn dsk oip osn sip. \\
  \text{rreq_rrep_sn (Rreq hops rreqid dip dsn dsk oip osn sip)} = \text{osn ≥ 1} \)"
"\(\forall hops dip dsn oip sip. \text{rreq_rrep_sn (Rrep hops dip dsn oip sip)} = \text{dsn ≥ 1} \)"
"\(\forall dests sip. \text{rreq_rrep_sn (Rerr dests sip)} = True\)"

30
"\∧\ d dip. \text{rreq\_rrep\_sn (Newpkt d dip) = True}\"

"\∧\ d dip sip. \text{rreq\_rrep\_sn (Pkt d dip sip) = True}\"

unfolding \text{rreq\_rrep\_sn\_def} by simp\_all

definition \text{rreq\_rrep\_fresh} :: \text{rt} ⇒ \text{msg} ⇒ \text{bool}

where "\text{rreq\_rrep\_fresh} crt m \equiv \text{case m of Rreq hopsc _ _ _ _ oipc osnc ipcc ⇒ (ipcc ≠ oipc ⇒ oipc}\notin\mathcal{D}(crt) \∧ (sqn crt oipc > osnc}
\lor (sqn crt oipc = osnc
\land the (dhops crt oipc) ≤ hopsc
\land the (flag crt oipc) = val)))}
\lor Rrep hopsc dip dsn dipc dsnc _ ipcc ⇒ (ipcc ≠ dipc ⇒ dipc}\notin\mathcal{D}(crt)
\land sqn crt dipc = dsnc
\land the (dhops crt dipc) = hopsc
\land the (flag crt dipc) = val)"

\land _ ⇒ True"

lemma \text{rreq\_rrep\_fresh [simp]}:

"\∧\ hops rreqid dip dsn dsnc dipc osnc sip.
\text{rreq\_rrep\_fresh} crt (Rreq hops rreqid dip dsn dsnc dipc osnc sip) =
(sip ≠ oip ⇒ dipc}\notin\mathcal{D}(crt)
\land (sqn crt oip > osnc
\lor (sqn crt oip = osnc
\land the (dhops crt oip) ≤ hopsc
\land the (flag crt oip) = val))"

"\∧\ hops dip dsn oip osnc sip.
\text{rreq\_rrep\_fresh} crt (Rreq hops dip dsn oip osnc sip) =
(sip ≠ dip ⇒ dipc}\notin\mathcal{D}(crt)
\land sqn crt dipc = dsnc
\land the (dhops crt dipc) = hopsc
\land the (flag crt dipc) = val)"

"\∧\ dests sip.
\text{rreq\_rrep\_fresh} crt (Rrep dests sip) = True"

"\∧\ d dip.
\text{rreq\_rrep\_fresh} crt (Newpkt d dip) = True"

"\∧\ d dip sip.
\text{rreq\_rrep\_fresh} crt (Pkt d dip sip) = True"

unfolding \text{rreq\_rrep\_fresh\_def} by simp\_all

definition \text{rerr\_invalid} :: "rt ⇒ msg ⇒ bool"

where "\text{rerr\_invalid} crt m \equiv \text{case m of Rerr destd dipc osnc rips ⇒ (∀ ripc∈\mathcal{D}\text{destsc}.}
\lor ripc\in\mathcal{D}(crt) \land the (destsc ripc) = sqn crt ripc))"

\land _ ⇒ True"

lemma \text{rerr\_invalid [simp]}:

"\∧\ hops rreqid dip dsn dsnc dipc osnc sip.
\text{rerr\_invalid} crt (Rreq hops rreqid dip dsn dsnc dipc osnc sip) = True"

"\∧\ hops dip dsn oip osnc sip.
\text{rerr\_invalid} crt (Rreq hops dip dsn oip osnc sip) = True"

"\∧\ dests sip.
\text{rerr\_invalid} crt (Rerr dests sip) = (∀ ripc∈\mathcal{D}(destsc).
ripc\in\mathcal{D}(crt) \land the (destsc ripc) = sqn crt ripc)"

"\∧\ d dip.
\text{rerr\_invalid} crt (Newpkt d dip) = True"

"\∧\ d dip sip.
\text{rerr\_invalid} crt (Pkt d dip sip) = True"

unfolding \text{rerr\_invalid\_def} by simp\_all

definition \text{init\_missing} :: (nat ⇒ state option) × \text{a} ⇒ (nat ⇒ state) × \text{a}

where "\text{init\_missing} σ = (λ i. case (fst σ) i of None ⇒ aodv\_init i | Some s ⇒ s, snd σ)"

lemma \text{not\_in\_net\_ips\_fst\_init\_missing [simp]}:

"\text{fst (init\_missing (net\_map \text{fst} σ)) i = aodv\_init i | Some s ⇒ s, snd σ)"

assumes "\text{fst (init\_missing (net\_map \text{fst} σ)) i = aodv\_init i}"

using assms unfolding \text{init\_missing\_def} by simp

lemma \text{fst\_init\_missing\_net\_map\_pair\_fst [simp]}:

"\text{fst (init\_missing (net\_map (\lambda (p, q). (\text{fst (id p), snd (id p), q)) s))}
= \text{fst (init\_missing (net\_map \text{fst} s))}"

unfolding \text{init\_missing\_def} by auto
We introduce a streamlined alternative to \textit{initmissing} with \textbf{netgmap} to simplify invariant statements and thus facilitate their comprehension and presentation.

\begin{verbatim}
lemma \textit{fst\_initmissing\_netgmap\_default\_aodv\_init\_netlift}:
  "\textit{fst (initmissing (netgmap \textit{fst\_s})) = default aodv\_init (netlift \textit{fst\_s})}"
unfolding \textit{initmissing\_def default\_def}
by (simp add: \textit{fst\_netgmap\_netlift del: One\_nat\_def})
\end{verbatim}

definition\textit{ netglobal} :: "((\textit{nat \Rightarrow state}) \Rightarrow \textit{bool}) \Rightarrow ((\textit{state} \times \textit{b}) \times \textit{c}) \textit{net\_state} \Rightarrow \textit{bool}"
where
  "\textit{netglobal} \textit{P} \equiv (\lambda s. \textit{P} \textit{(default aodv\_init (netlift \textit{fst\_s}))})"
end

0.6 Quality relations between routes

theory \textit{Fresher}
imports \textit{Aodv\_Data}
begin

0.6.1 Net sequence numbers

On individual routes

\begin{verbatim}
definition \textit{nsqn} :: "\textit{r} \Rightarrow \textit{sqn}"
where
  "\textit{nsqn} \textit{r} \equiv (if \pi 4 \textit{r} = \textit{val} \lor \pi 2 \textit{r} = 0 then \pi 2 \textit{r} else (\pi 2 \textit{r} - 1))"
\end{verbatim}

\begin{verbatim}
lemma \textit{nsqnr\_def}':
  "\textit{nsqn} \textit{r} = (if \pi 4 \textit{r} = \textit{inv} then \pi 2 \textit{r} - 1 else \pi 2 \textit{r})"
unfolding \textit{nsqn\_def} by simp
\end{verbatim}

\begin{verbatim}
lemma \textit{nsqn\_zero [simp]}:
  "\textit{sqn} \textit{r} (0, \textit{dsk}, \textit{flag}, \textit{hops}, \textit{nhip}, \textit{pre}) = 0"
unfolding \textit{nsqn\_def} by clarsimp
\end{verbatim}

\begin{verbatim}
lemma \textit{nsqn\_val [simp]}:
  "\textit{sqn} \textit{r} (\textit{dsn}, \textit{dsk}, \textit{hops}, \textit{nhip}, \textit{pre}) = \textit{dsn}"
unfolding \textit{nsqn\_def} by clarsimp
\end{verbatim}

\begin{verbatim}
lemma \textit{nsqn\_inv [simp]}:
  "\textit{sqn} \textit{r} (\textit{dsn}, \textit{dsk}, \textit{inv}, \textit{hops}, \textit{nhip}, \textit{pre}) = \textit{dsn} - 1"
unfolding \textit{nsqn\_def} by clarsimp
\end{verbatim}

\begin{verbatim}
lemma \textit{nsqn\_lte\_dsn [simp]}:
  "\textit{sqn} \textit{r} (\textit{dsn}, \textit{dsk}, \textit{flag}, \textit{hops}, \textit{nhip}, \textit{pre}) \leq \textit{dsn}"
unfolding \textit{nsqn\_def} by clarsimp
\end{verbatim}

On routes in routing tables

\begin{verbatim}
definition \textit{nsqn} :: "\textit{rt} \Rightarrow \textit{ip} \Rightarrow \textit{sqn}"
where
  "\textit{nsqn} \equiv (\lambda \textit{rt} \textit{dip}. \textit{case } \sigma_{\textit{route}} (\textit{rt}, \textit{dip}) \textit{ of None } \Rightarrow 0 | \textit{Some } \textit{r} \Rightarrow \textit{nsqn}\_\textit{r} (\textit{r}))"
\end{verbatim}

\begin{verbatim}
lemma \textit{nsqn\_sqn\_def}:
  "\textit{sqn} \textit{r} \textit{dip} = (if \textit{flag} \textit{rt} \textit{dip} = \textit{Some } \textit{val} \lor \textit{sqn} \textit{rt} \textit{dip} = 0
  then \textit{sqn} \textit{rt} \textit{dip} else \textit{sqn} \textit{rt} \textit{dip} - 1)"
unfolding \textit{nsqn\_def sqn\_def} by (clarsimp split: option.split) auto
\end{verbatim}

\begin{verbatim}
lemma \textit{not\_in\_kD\_nsqn [simp]}:
  assumes "\textit{dip} \notin \textit{kD}\_\textit{rt}"
  shows "\textit{nsqn} \textit{rt} \textit{dip} = 0"
\end{verbatim}

32
using \textit{assms unfolding \textit{nsqn} def} by \textit{simp}

\textbf{lemma \textit{kD} \textit{nsqn}:}
  \textit{assms "dip \in kD(rt)"}
  \textit{shows "nsqn rt dip = nsqn rt (the (\sigma_{route}(rt, dip)))"}
  \textit{using \textit{assms [THEN kD\_Some] unfolding \textit{nsqn} def by clarsimp}

\textbf{lemma \textit{nsqnr\_r\_flag\_pred} [simp, intro]:}
  \textit{fixes dsn dsk flag hops nhip pre}
  \textit{assumes "P (nsqn rt (dsn, dsk, val, hops, nhip, pre))"}
  \textit{and "P (nsqn rt (dsn, dsk, inv, hops, nhip, pre))"}
  \textit{shows "P (nsqn rt (dsn, dsk, flag, hops, nhip, pre))"}
  \textit{using \textit{assms by (cases flag) auto}

\textbf{lemma \textit{nsqn\_\_addpreRT\_inv} [simp]:}
  \textit{"\forall dip npre dip'. dip \in kD(rt) \implies nsqn rt (the (the (addpreRT rt dip npre) dip')) = nsqn rt (the (rt dip'))"}
  \textit{unfolding addpreRT\_def nsqn\_\_def}
  \textit{by (frule kD\_Some) (clarsimp split: option.split)

\textbf{lemma \textit{sqn\_nsqn}:}
  \textit{"\forall rt dip. sqn rt dip - 1 \leq nsqn rt dip"}
  \textit{unfolding sqn\_def nsqn\_def by (clarsimp split: option.split)

\textbf{lemma \textit{val\_nsqn\_sqn} [elim, simp]:}
  \textit{assumes "ip\in kD(rt)"}
  \textit{and "the (flag rt ip) = val"}
  \textit{shows "nsqn rt ip = sqn rt ip"}
  \textit{using \textit{assms unfolding \textit{nsqn\_sqn} def by auto

\textbf{lemma \textit{vd\_nsqn\_sqn} [elim, simp]:}
  \textit{assumes "ip\in vD(rt)"}
  \textit{shows "nsqn rt ip = sqn rt ip"}
  \textit{proof -
      from (ip\in vD(rt)) have "ip\in kD(rt)"
      and "the (flag rt ip) = val" by auto
      thus ?thesis ..
  qed

\textbf{lemma \textit{inv\_nsqn\_sqn} [elim, simp]:}
  \textit{assumes "ip\in kD(rt)"}
  \textit{and "the (flag rt ip) = inv"}
  \textit{shows "nsqn rt ip = sqn rt ip - 1"}
  \textit{using \textit{assms unfolding \textit{nsqn\_sqn} def by auto

\textbf{lemma \textit{id\_nsqn\_sqn} [elim, simp]:}
  \textit{assumes "ip\in iD(rt)"}
  \textit{shows "nsqn rt ip = sqn rt ip - 1"}
  \textit{proof -
      from (ip\in iD(rt)) have "ip\in kD(rt)"
      and "the (flag rt ip) = inv" by auto
      thus ?thesis ..
  qed

\textbf{lemma \textit{nsqn\_update\_changed\_kno\_val} [simp]: "\forall rt dip dsn dsk hops nhip.}
  \textit{rt \neq update rt ip (dsn, kno, val, hops, nhip, {})}
  \textit{\implies nsqn (update rt ip (dsn, kno, val, hops, nhip, {})) ip = dsn"}
  \textit{using \textit{assms unfolding \textit{nsqn\_\_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_.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)

0.6.2 Comparing routes

definition
fresher :: "r ⇒ r ⇒ bool" ("(_/ ⊑ _)") [51, 51] 50
where
"fresher r r’ ≡ ((nsqn r < nsqn r’) ∨ (nsqn r = nsqn r’ ∧ π₅(r) ≥ π₅(r’)))"

lemma fresherI1 [intro]:
assumes "nsqn r < nsqn r’"
shows "r ⊑ r’"
unfolding fresher_def using assms by simp

lemma fresherI2 [intro]:
assumes "nsqn r = nsqn r’" 
and "π₅(r) ≥ π₅(r’)"
shows "r ⊑ r’"
unfolding fresher_def using assms by simp

lemma fresherI [intro]:
assumes "(nsqn r < nsqn r’) ∨ (nsqn r = nsqn r’ ∧ π₅(r) ≥ π₅(r’))"
shows "r ⊑ r’"
unfolding fresher_def using assms .

lemma fresherE [elim]:
assumes "r ⊑ r’"
and "nsqn r < nsqn r’ ⇒ P r r’"
and "nsqn r = nsqn 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

0.6.3 Comparing routing tables

definition
  rt_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
where
  "rt_fresher ≡ λdip rt rt’. (the (σ_route(rt, dip))) ⊑ (the (σ_route(rt’, dip)))"

abbreviation
  rt_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_/ ⊑ _/ _)"
where
  "rt1 ⊑ i rt2 ≡ rt_fresher i rt1 rt2"

lemma rt_fresher_def':
  "(rt1 ⊑ rt2) = (nsqn rt1 dip < nsqn rt2 dip ∨
    nsqn rt1 dip = nsqn rt2 dip ∧ π₅ (the (dhops rt1 dip)) ≥ π₅ (the (dhops rt2 dip)))"
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"

35
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 (\lambda ip. 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 ≡ \lambda dip rt1 rt2. (rt1 ⊑_dip rt2) ∧ (rt2 ⊑_dip rt1)"

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

lemma rt_fresh_as_refl [simp]: "\(rt \Rightarrow= \Rightarrow rt \Rightarrow bool" ("\_/≈ \_/") [51, 999, 51] 50
unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
"\(rt1 \Rightarrow= dip rt2; rt2 \Rightarrow= dip rt3 \] ⇒ rt1 \Rightarrow= 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) ⊑ the (rt2 dip)"
and "the (rt2 dip) ⊑ 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 "\(\pi_5(\text{the (rt dip)}) = \pi_5(\text{the (rt' dip)})\)"
shows "rt \textsubscript{\text{dip}} \approx 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 \textsubscript{\text{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' \textsubscript{\text{dip}} rt"
  by (rule rt_fresherI2)

qed

lemma rt_fresh_asE [elim]:
  assumes "rt1 \approx dip rt2"
  and "[ rt1 \textsubscript{\text{dip}} rt2; rt2 \textsubscript{\text{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 \textsubscript{\text{dip}} rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
  assumes "rt1 \approx dip rt2"
  shows "rt2 \textsubscript{\text{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 \textsubscript{\text{dip}} rt2)"
  shows "\neg (rt1 \approx dip rt2)"
proof
  assume "rt1 \approx dip rt2"
  hence "rt1 \textsubscript{\text{dip}} rt2."..
  with \neg (rt1 \textsubscript{\text{dip}} rt2) show False ..

qed

lemma not_rt_fresh_asI2 [intro]:
  assumes "\neg (rt2 \textsubscript{\text{dip}} rt1)"
  shows "\neg (rt1 \approx dip rt2)"
proof
  assume "rt1 \approx dip rt2"
  hence "rt2 \textsubscript{\text{dip}} rt1."..
  with \neg (rt2 \textsubscript{\text{dip}} rt1) show False ..

qed

lemma not_single_rt_fresher [elim]:
  assumes "\neg (\text{the (rt1 ip)} \subseteq \text{the (rt2 ip)})"
  shows "\neg (rt1 \textsubscript{\text{ip}} rt2)"
proof
  assume "rt1 \textsubscript{\text{ip}} rt2"
  hence "the (rt1 ip) \subseteq the (rt2 ip)."..
  with \neg (the (rt1 ip) \subseteq 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 "nsqn r (the (rt2 dip)) = nsqn r (the (rt1 dip))"
using assms(3) unfolding rt_fresher_def
by (auto simp: rt_fresher_def2 [OF ⟨dip ∈ kD(rt1)⟩ ⟨dip ∈ kD(rt2)⟩]
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 "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 dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n
where rtnt [simp]: "the (rt dip) = (dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)"
by (metis prod_cases6)
with ⟨the (dhops rt dip) ≥ 1⟩ and ⟨dip ∈ kD(rt)⟩ have "hops_n ≥ 1"
by (metis proj5_eq_dhops proj6(4))
from ⟨dip ∈ kD(rt)⟩ rtnt have [simp]: "sqn rt dip = dsn_n"
  and [simp]: "the (dhops rt dip) = hops_n"
  and [simp]: "the (flag rt dip) = f_n"
by (simp add: sqn_def proj5_eq_dhops [symmetric]
proj4_eq_flag [symmetric])

from ⟨update_arg_wf r⟩ have "(dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)"
proof (rule wf_r_cases)
fix nip pre
from \( \langle \text{hops}_n \geq 1 \rangle \) have "\( \langle \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n \rangle \)\n\( \subseteq \langle \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n \rangle \)"
unfolding fresher_def sqn_def by (cases \( f_n \)) auto
thus "\( \langle \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n \rangle \)\n\( \subseteq \langle \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n \rangle \)"
unfolding fresher_def sqn_def by (cases \( f_n \)) auto
next
fix dsn :: sqn
and hops nip pre
assume "0 < dsn" show "\( \langle \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n \rangle \)\n\( \subseteq \langle \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n, \text{dsn}_n \rangle \)"
proof (rule update_cases_kD [OF _ \( \langle \text{dip} \in \text{kD}(rt) \rangle \)], simp_all add: \( \langle 0 < dsn \rangle \))
assumption
ultimately show ?thesis unfolding fresher_def nsqn_r_def by simp
next
assume "dsn_n = dsn" with \( \langle 0 < dsn \rangle \) show ?thesis unfolding fresher_def by simp
qed
hence "rt \( \subseteq \text{dip} \) update rt dip r" by - (rule single_rt_fresher, simp)
with \( \langle \text{dip} = \text{ip} \rangle \) show ?thesis by simp
qed

theorem rt_fresher.invalidate [simp]:
assumes "\( \text{dip} \in \text{kD}(rt) \)"
and indests: "\( \forall \text{rip} \in \text{dom}(\text{dests}) \). \text{rip} \in \text{vD}(rt) \wedge \text{sqn}(rt) \text{ rip} < \text{the}(\text{dests rip}) \)"
shows "rt \( \subseteq \text{dip} \) invalidate rt dests"
proof (cases "\( \text{dip} \in \text{dom}(\text{dests}) \)"
assume "\( \text{dip} \in \text{dom}(\text{dests}) \)"
thus ?thesis using \( \langle \text{dip} \in \text{kD}(rt) \rangle \)
proof (cases "\( \text{dip} \in \text{dom}(\text{dests}) \)"
assumption
ultimately show ?thesis unfolding invalidate_def sqn_def
by - (rule single_rt_fresher, auto simp: fresher_def)
qed

lemma nsqn_r.invalidate [simp]:
assumes "\( \text{dip} \in \text{kD}(rt) \)"
and \( \text{dip} \in \text{dom}(\text{dests}) \) shows "\( \text{nsqn_r}(\text{the}(\text{invalidate rt dests dip})) = \text{the}(\text{dests dip}) - 1 \)"
using assms unfolding invalidate_def by auto

lemma rt_fresh_as_inc.invalidate [simp]:
assumes "\( \text{dip} \in \text{kD}(rt) \)"
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 ⟨dip∈kD(rt)⟩ have "dip∈kD(invalidate rt dests)"
by simp
moreover then have "dip∈kD(invalidate rt dests)" by simp
ultimately show thesis
proof (rule nsqn_rt_fresh_asI)
from ⟨dip∈kD(rt)⟩ have "nsqn rt dip = sqn rt dip" by simp
also have "sqn rt dip = nsqn (invalidate rt dests dip)"
proof -
from ⟨dip∈kD(rt)⟩ have "nsqn (the (invalidate rt dests dip)) = the (dests dip) - 1" using ⟨dip∈dom(dests)⟩
by (rule nsqn_invalidate)
with ⟨the (dests dip) = inc (sqn rt dip)⟩ show "sqn rt dip = nsqn (the (invalidate rt dests dip))" by simp
qed
also from ⟨dip∈kD(invalidate rt dests)⟩ have "nsqn (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) ∧ ¬(rt2 ⊑ dip rt1)"

abbreviation rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("(_ ⊣⊢ _)" [51, 999, 51] 50)
where "rt1 ⊣⊢ rt2 ≡ rt_strictly_fresher i rt1 rt2"

lemma rt_strictly_fresher_def'':
"rt1 ⊣⊢ rt2 = ((rt1 ⊣⊢ rt2) ∧ ¬(rt2 ⊣⊢ rt1))"
unfolding rt_strictly_fresher_def rt_fresh_as_def by auto

lemma rt_strictly_fresherI' [intro]:
assumes "rt1 ⊣⊢ rt2"
and "¬(rt2 ⊣⊢ rt1)"
shows "rt1 ⊣⊢ rt2"
using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherE' [elim]:
assumes "rt1 ⊣⊢ rt2"
and "[ rt1 ⊣⊢ rt2; ¬(rt2 ⊣⊢ 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 ⊑ rt2"
and "¬(rt1 ≈ rt2)"
shows "rt1 ⊏ 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 ⊏ rt2"
and "[ rt1 ⊑ rt2; ¬(rt1 ≈ 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 ⊏ rt2 =
(nsqn_r (the (rt1 i)) < nsqn_r (the (rt2 i))
∨ (nsqn_r (the (rt1 i)) = nsqn_r (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 ⊏ dip rt2"
shows "the (rt1 dip) ⊑ 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 ⊏ dip rt2"
shows "¬ rt1 ≈ dip rt2"
using assms unfolding rt_strictly_fresher_def
by auto

lemma rt_strictly_fresher_trans [elim, trans]:
assumes "rt1 ⊑ dip rt2"
and "rt2 ⊑ dip rt3"
shows "rt1 ⊑ dip rt3"
using assms proof -
from rt1 ⊑ dip rt2. obtain "the (rt1 dip) ⊑ the (rt2 dip)" by auto
also from rt2 ⊑ dip rt3 obtain "the (rt2 dip) ⊑ the (rt3 dip)" by auto
finally have "the (rt1 dip) ⊑ the (rt3 dip)" .

moreover have "¬ (rt1 ≈ dip rt3)"
proof -
from rt1 ⊑ dip rt2. obtain "¬(the (rt2 dip) ⊑ the (rt1 dip))" by auto
also from rt2 ⊑ dip rt3. obtain "¬(the (rt3 dip) ⊑ the (rt2 dip))" by auto
finally have "¬(the (rt3 dip) ⊑ the (rt1 dip))" .
thus ?thesis ..
qed ultimately show "rt1 ⊑ dip rt3" ..

qed

lemma rt_strictly_fresher_irefl [simp]: "¬ (rt ⊑ dip rt)"
using assms unfolding rt_strictly_fresher_def
by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
assumes "rt1 ⊑ dip rt2"
and "rt2 ⊑ dip rt3"
shows "rt1 ⊑ dip rt3"
proof -
from rt1 ⊑ dip rt2. have "rt1 ⊑ dip rt2"
and "¬(rt2 ⊑ dip rt1)"


unfolding \texttt{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 \texttt{rt\_strictly\_fresher\_def''} by auto
qed

lemma \texttt{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 \texttt{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 \texttt{rt\_strictly\_fresher\_def''} by auto
qed

lemma \texttt{rt\_fresher\_imp\_nsqn\_le}:
assumes "rt1 \sqsubseteq dip 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: \texttt{rt\_fresher\_def2} [OF assms(2-3)])

lemma \texttt{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 \texttt{rt\_fresher\_imp\_nsqn\_le})
with \langle nsqn rt1 dip < nsqn rt2 dip \rangle show "False"
by simp
qed

lemma \texttt{rt\_strictly\_fresher\_eqI} [intro]:
assumes "i \in kD(rt1)"
and "i \in kD(rt2)"
and \( \pi_j (\text{rt}_1 i) = \pi_j (\text{rt}_2 i) \)
and \( \pi_j (\text{rt}_2 i) < \pi_j (\text{rt}_1 i) \)
shows \( \text{rt}_1 \sqsubseteq_i \text{rt}_2 \)
using \text{assms unfolding \text{rt}_strictly\_fresher_def'} by (auto simp add: kD_nsqn)

**lemma invalidate\_rtsf\_left [simp]:**
\[
\forall \text{dests dip r r'}. \text{dests dip = None} \implies (\text{invalidate rt dests} \sqsupset dip r') = (rt \sqsupset dip r')
\]
unfolding invalidate\_def rt\_strictly\_fresher_def'
by (rule iffI) (auto split: option.split_asm)

**lemma vD\_invalidate\_rt\_strictly\_fresher [simp]:**
assumes \( \text{dip} \in \text{vD}(\text{invalidate rt1 dests}) \)
shows \( (\text{invalidate rt1 dests} \sqsupset dip \text{rt2}) = (\text{rt1} \sqsupset dip \text{rt2}) \)
proof (cases \( \text{dip} \in \text{dom(dests)} \))
assume \( \text{dip} \notin \text{dom(dests)} \) hence \( \text{dip} \notin \text{vD}(\text{invalidate rt1 dests}) \)
unfolding invalidate\_def vD\_def
by clarsimp (metis \text{assms option.simps(3)} vD\_invalidate\_vD\_not\_dests)
next
assume \( \text{dip} \in \text{dom(dests)} \) hence \( \text{dip} \in \text{vD}(\text{invalidate rt1 dests}) \)
unfolding invalidate\_def rt\_strictly\_fresher_def'
by auto

**lemma rt\_strictly\_fresher\_update\_other [elim!]:**
\[
\forall \text{dip ip r r'}. [ \text{dip} \neq \text{ip}; \text{rt} \sqsupset dip \text{rt'} ] \implies \text{update rt ip r} \sqsupset dip \text{rt'}
\]
unfolding rt\_strictly\_fresher_def'
by clarsimp

**lemma addpreRT\_strictly\_fresher [simp]:**
assumes \( \text{dip} \in \text{kD}(\text{rt}) \)
shows \( (\text{addpreRT rt dip npre}) \sqsupset dip \text{rt2} = (\text{rt} \sqsupset dip \text{rt2}) \)
using \text{assms unfolding \text{rt}_strictly\_fresher_def'} by clarsimp

**lemma lt\_sqn\_imp\_update\_strictly\_fresher:**
assumes \( \text{dip} \in \text{vD}(\text{rt2 nhip}) \)
and \*: \( \text{osn} < \text{sqn} (\text{rt2 nhip} \text{ dip}) \)
and **: \( \text{rt} \neq \text{update rt dip (osn, kno, val, hops, nhip, {})} \)
shows \( \text{update rt dip (osn, kno, val, hops, nhip, {})} \sqsupset dip \text{rt2 nhip} \)
unfolding rt\_strictly\_fresher_def'
proof (rule disjI1)
from ** have \( \text{nsqn} (\text{update rt dip (osn, kno, val, hops, nhip, {})} \text{ dip}) = \text{osn} \)
by (rule nsqn\_update\_changed\_kno\_val)
with \( \text{dip} \in \text{vD}(\text{rt2 nhip}) \)
have \( \text{nsqn} (\text{the (update rt dip (osn, kno, val, hops, nhip, {})} \text{ dip})) = \text{osn} \)
by (simp add: kD_nsqn)
also have \( \text{osn} < \text{sqn} (\text{rt2 nhip} \text{ dip}) \) by (rule *)
also have \( \text{sqn} (\text{rt2 nhip} \text{ dip}) = \text{nsqn} (\text{the (rt2 nhip dip)}) \)
unfolding nsqn\_def using \( \text{dip} \in \text{vD}(\text{rt2 nhip}) \)
by - (metis vD\_flag\_val proj2_eq_sqn proj4_eq_flag vD\_id\_gives\_kD(1))
finally show \( \text{nsqn} (\text{the (update rt dip (osn, kno, val, hops, nhip, {})} \text{ dip})) < \text{nsqn} (\text{the (rt2 nhip dip)}) \).

**qed**

**lemma dhops\_le\_hops\_imp\_update\_strictly\_fresher:**
assumes \( \text{dip} \in \text{vD}(\text{rt2 nhip}) \)
and \( \text{sqn} \): \( \text{sqn} (\text{rt2 nhip} \text{ dip}) = \text{osn} \)
and **: \( \text{rt} \neq \text{update rt dip (osn, kno, val, Suc hops, nhip, {})} \)
shows \( \text{update rt dip (osn, kno, val, Suc hops, nhip, {})} \sqsupset dip \text{rt2 nhip} \)

43
unfolding \textit{rt\_strictly\_fresher\_def}'
proof (rule disjI2, rule conjI)
from ** have "\textit{nsqn} (update \textit{rt} dip (\textit{osn}, \textit{kno}, \textit{val}, Suc \textit{hops}, \textit{nhip}, \{\}) dip = \textit{osn}"
  by (rule \textit{nsqn\_update\_changed\_kno\_val})
with (\textit{dip} \in \textit{vD}(\textit{rt2 nhip}):
  have "\textit{nsqn}, (the (update \textit{rt} dip (\textit{osn}, \textit{kno}, \textit{val}, Suc \textit{hops}, \textit{nhip}, \{\}) dip)) = \textit{osn}"
by (simp add: kD_nsqn)
also have "\textit{osn} = \textit{sqn} (\textit{rt2 nhip}) dip" by (rule \textit{sqn [symmetric]})
also have "\textit{sqn} (\textit{rt2 nhip}) dip = \textit{nsqn}, (the (\textit{rt2 nhip} dip))"
  unfolding \textit{nsqn}, _def using (\textit{dip} \in \textit{vD}(\textit{rt2 nhip}):
finally show "\textit{nsqn}, (the (update \textit{rt} dip (\textit{osn}, \textit{kno}, \textit{val}, Suc \textit{hops}, \textit{nhip}, \{\}) dip))
  = \textit{nsqn}, (the (\textit{rt2 nhip} dip))".
next
  have "the (\textit{dhops} (\textit{rt2 nhip}) dip) \leq \textit{hops}" by (rule \textit{hop})
  also have "\textit{hops} < \textit{hops} + 1" by simp
  finally have "the (\textit{dhops} (\textit{rt2 nhip}) dip)
    < the (\textit{dhops} (update \textit{rt} dip (\textit{osn}, \textit{kno}, \textit{val}, Suc \textit{hops}, \textit{nhip}, \{\}) dip))"
    using \textit{dip} \in \textit{vD}(\textit{rt2 nhip}) by (simp add: proj5_eq_dhops)
  thus "\pi_5 (the (\textit{rt2 nhip} dip)) < \pi_5 (the (update \textit{rt} dip (\textit{osn}, \textit{kno}, \textit{val}, Suc \textit{hops}, \textit{nhip}, \{\}) dip))"
  by (simp add: proj5_eq_dhops)
  qed

lemma \textit{nsqn\_invalidate}:
  assumes "\textit{dip} \in \textit{kD}(\textit{rt})"
  and "\forall \textit{ip} \in \textit{dom}(\textit{dests}). \textit{ip} \in \textit{vD}(\textit{rt}) \wedge \textit{the} (\textit{dests} \textit{ip}) = \textit{inc} \ (\textit{sqn} \textit{rt} \textit{ip})"
  shows "\textit{nsqn} (\textit{invalidate} \textit{rt} \textit{dests}) \textit{dip} = \textit{nsqn} \textit{rt} \textit{dip}"
proof -
  from assms have "\textit{rt} \approx_{\textit{dip}} \textit{invalidate} \textit{rt} \textit{dests}" by simp
  also have "\textit{rt} \approx_{\textit{dip}} \textit{invalidate} \textit{rt} \textit{dests}"
    by (rule \textit{rt\_fresh\_as\_inc\_invalidate})
  with (\textit{dip} \in \textit{kD}(\textit{rt}) \& \textit{dip} \in \textit{kD}(\textit{invalidate} \textit{rt} \textit{dests})) show \textit{thesis}
    by (simp add: kD_nsqn del: invalidate_kD_inv)
  (erule(2) \textit{rt\_fresh\_as\_nsqnrr})
  qed

end

\section{Invariant proofs on individual processes}

theory \textit{Seq\_Invariants}
imports "./\textit{AWN/Invariants} \textit{Aodv} \textit{Aodv\_Data} \textit{Aodv\_Predicates} \textit{Fresher}
begin

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

Proposition 7.2

lemma \textit{sequence\_number\_increases}:
  "\textit{paodv} \textit{i} \models \textit{A} \textit{onll} \Gamma_{\textit{AODV}} (\lambda((\xi, _), _, (\xi', _)). \textit{sn} \xi \leq \textit{sn} \xi')"
by \textit{inv\_cterms}

lemma \textit{sequence\_number\_one\_or\_bigger}:
  "\textit{paodv} i \models \textit{onl} \Gamma_{\textit{AODV}} (\lambda(\xi, _). \textit{1} \leq \textit{sn} \xi)"
by (rule \textit{onll\_step\_to\_invariantI} [OF \textit{sequence\_number\_increases}])
  (auto simp: \textit{\sigma}_{\textit{AODV\_def}})

We can get rid of the \textit{onl/onll} if desired...

lemma \textit{sequence\_number\_increases}':
  "\textit{paodv} i \models \textit{A} (\lambda(((\xi, _), _, (\xi', _)). \textit{sn} \xi \leq \textit{sn} \xi')"
by (rule \textit{step\_invariant\_weakenE} [OF \textit{sequence\_number\_increases}]) (auto dest!: onllD)
lemma sequence_number_one_or_bigger':
"paodv i \models (\lambda(\xi, \_). 1 \leq sn \xi)"
by (rule invariant_weakenE [OF sequence_number_one_or_bigger]) auto

lemma sip_in_kD:
"paodv i \models onl \Gamma_{\text{AODV}} (\lambda(\xi, 1). 1 \in \{\text{PAodv-:7} \cup \text{PAodv-:5} \cup \text{PRrep-:0..PRrep-:1} \cup \{\text{PRreq-:0..PRreq-:3}\} \longrightarrow sip \xi \in kD (rt \xi))"
by inv_cterms

lemma rrep_1_update_changes:
"paodv i \models onl \Gamma_{\text{AODV}} (\lambda(\xi, l). (l = \text{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:
"paodv i \models onl \Gamma_{\text{AODV}} (\lambda(\xi, l). (l \in \{\text{PRreq-:16..PRreq-:18} \cup \{\text{PRreq-:2..PRreq-:6}\} \longrightarrow dip \xi \in kD (rt \xi)) \land (l \in \{\text{PRreq-:3..PRreq-:17}\} \longrightarrow oip \xi \in kD (rt \xi)))"
by inv_cterms

Proposition 7.38
lemma includes_nhip:
"paodv i \models onl \Gamma_{\text{AODV}} (\lambda(\xi, l). \forall dip \in kD(rt \xi). \text{the (nhop (rt \xi) dip)}) \in kD(rt \xi))"
proof -
{ fix ip and \xi \xi' :: state
  assume "\forall dip\in kD (rt \xi). \text{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
    \lor 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). \text{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, {}))) dip = ip
    \lor the (nhop (update (rt \xi) ip (sn, kno, val, Suc hops, sip, {}))) dip \in kD (rt \xi))
    \land (\forall dip\in kD (rt \xi).
      the (nhop (update (rt \xi) ip (sn, kno, val, Suc hops, sip, {}))) dip = ip
      \lor 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 aodv_wf sip_in_kD]
  onl_invariant_sterms [OF aodv_wf addpreRT_partly_welldefined]
  solve: one_hop nhip_is_sip)
qed

Proposition 7.22: needed in Proposition 7.4
lemma addpreRT_welldefined:
"paodv i \models onl \Gamma_{\text{AODV}} (\lambda(\xi, l). (l \in \{\text{PRreq-:16..PRreq-:18} \cup \{\text{PRreq-:2..PRreq-:6}\} \longrightarrow dip \xi \in kD (rt \xi)) \land (l \in \{\text{PRreq-:3..PRreq-:17}\} \longrightarrow oip \xi \in kD (rt \xi)))"
(is "$_\models onl \Gamma_{\text{AODV}} ?P$")
unfolding invariant_def
proof -
fix s
assume "s \in reachable (paodv i) TT"
then obtain \xi p where "s = (\xi, p)"
  and "(\xi, p) \in reachable (paodv i) TT"
by (metis PairE)

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

Proposition 7.4

**lemma known_destinations_increase:**

"\( \text{paodv} \ i \|= \text{onl} \ \Gamma_{AODV} \ (\lambda(\xi, \_), \_, (\xi', \_)). \text{kD} \ (\text{rt} \ \xi) \subseteq \text{kD} \ (\text{rt} \ \xi') \)"

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]
  simp add: subset_insertI)

Proposition 7.5

**lemma rreqs_increase:**

"\( \text{paodv} \ i \|= \text{onl} \ \Gamma_{AODV} \ (\lambda(\xi, \_), \_, (\xi', \_)). \text{rreqs} \ \xi \subseteq \text{rreqs} \ \xi' \)"

by (inv_cterms simp add: subset_insertI)

**lemma dests_bigger_than_sqn:**

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

proof -
  have sqninv:
    "\( \forall \text{dests} \ \text{rt} \ \text{rsn} \ \text{ip}. \ [ \ \forall \text{ip} \in \text{dom}(\text{dests}). \ \text{ip} \in \text{kD}(\text{rt}) \land \text{sqn} \ \text{rt} \ \text{ip} \leq \text{the} \ (\text{dests} \ \text{ip}) \ ]; \ \text{dests} \ \text{ip} = \text{Some} \ \text{rsn} \ ]
    \rightarrow \text{sqn} \ (\text{invalidate} \ \text{rt} \ \text{dests}) \ \text{ip} \leq \text{rsn}\)"
    by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto
  have indests:
    "\( \forall \text{dests} \ \text{rt} \ \text{rsn} \ \text{ip}. \ [ \ \forall \text{ip} \in \text{dom}(\text{dests}). \ \text{ip} \in \text{kD}(\text{rt}) \land \text{sqn} \ \text{rt} \ \text{ip} \leq \text{the} \ (\text{dests} \ \text{ip}) \ ]; \ \text{dests} \ \text{ip} = \text{Some} \ \text{rsn} \ ]
    \rightarrow \text{ip} \in \text{kD}(\text{rt}) \land \text{sqn} \ \text{rt} \ \text{ip} \leq \text{rsn}\)"
    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:**

"\( \text{paodv} \ i \|= \text{onl} \ \Gamma_{AODV} \ (\lambda(\xi, \_). \ \forall \text{ip}. \ \text{sqn} \ (\text{rt} \ \xi) \ \text{ip} \leq \text{sqn} \ (\text{rt} \ \xi') \ \text{ip}) \)"

proof -
  { fix \( \xi \) :: state
    assume \*: "\( \forall \text{ip} \in \text{dom}(\text{dests} \ \xi). \ \text{ip} \in \text{kD} \ (\text{rt} \ \xi) \land \text{sqn} \ (\text{rt} \ \xi) \ \text{ip} \leq \text{the} \ (\text{dests} \ \xi \ \text{ip}) \)"
    have "\( \forall \text{ip}. \ \text{sqn} \ (\text{rt} \ \xi) \ \text{ip} \leq \text{sqn} \ (\text{invalidate} \ (\text{rt} \ \xi) \ (\text{dests} \ \xi)) \ \text{ip})"\)
    proof
      fix \( \text{ip} \)
**Proposition 7.7**

**lemma ip_constant:**

"paodv i |= onl Γ_{AODV} (λ(ξ, _). ip ξ = i)"

by (inv_cterms simp add: σ_{AODV}_def)

**Proposition 7.8**

**lemma sender_ip_valid':**

"paodv i |=A onll Γ_{AODV} (λ(((ξ, _), a, _). anycast (λm. not_Pkt m → msg_sender m = ip ξ)) a)"

by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid'])

(auto dest!: onlD onllD)

**lemma sender_ip_valid:**

"paodv i |=A onll Γ_{AODV} (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)

**Proposition 7.9**

**lemma received_msg_inv:**

"paodv i |= (recvmsg P → P (msg ξ))"

by inv_cterms

**Proposition 7.10**

**lemma hop_count_positive:**

"paodv i |= onl Γ_{AODV} (λ(ξ, _). ∀ip∈kD (rt ξ). the (dhops (rt ξ) ip) ≥ 1)"

by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined])

**lemma rreq_dip_in_vD_dip_eq_ip:**

"paodv i |= onl Γ_{AODV} (λ(ξ, 1). 1 ∈ {PRreq-:16..PRreq-:18} → dip ξ ∈ vD(rt ξ))
∧ (1 ∈ {PRreq-:5, PRreq-:6} → dip ξ = ip ξ)
∧ (1 ∈ {PRreq-:15..PRreq-:18} → dip ξ ≠ ip ξ))"

**proof**

(auto dest: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined])

47
with \( \text{dip } \xi \in \text{vD } \text{(rt } \xi ) \);

show "dip \( \xi \in \text{vD } \text{(the } (\text{addpreRT } \text{(rt } \xi ) \text{(oip } \xi ) \text{the } (\text{nhop } \text{(rt } \xi ) \text{(dip } \xi )))\))" by simp

qed

Proposition 7.11

lemma anycast_msg_zhops:
"\( \forall \text{rreqid dip dsn dsk oip osn sip.} \)
\( \text{paodv } i =\rightarrow \text{ onll } \Gamma_{\text{AODV }} (\lambda (\zeta, a, \ldots). \text{anycast msg_zhops a}) \)"

proof (inv cterms inv add:
\( \text{onl_invariant_sterms } \text{[OF aodv_wf rreq_dip_in_vD_dip_eq_ip [THEN invariant_restrict_inD]]} \)
\( \text{onl_invariant_sterms } \text{[OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]]}, \)
elim conjE)

fix \( l \xi a pp p' pp' \)
assume
\( (\xi, pp) \in \text{reachable } \text{(paodv } i ) \text{ TT} \)
and
\(" PRreq-:18} \text{unicast}(\lambda \xi. \text{the } (\text{nhop } \text{(rt } \xi ) \text{(oip } \xi )),(\lambda \xi. \text{Rrep } (\text{the } (\text{dhops } \text{(rt } \xi ) \text{(dip } \xi ))),(\lambda \xi. \text{sqn } (\text{rt } \xi ) \text{(dip } \xi )),(\lambda \xi. \text{oip } \xi )),(\lambda \xi. \text{ip } \xi ))) \) .
\( \text{p' } \triangleright pp' \in \text{sterms } \Gamma_{\text{AODV }} pp \)
and
\( l = PRreq-:18 \)
and
\(" a = \text{unicast } (\text{the } (\text{nhop } \text{(rt } \xi ) \text{(oip } \xi ))) (\text{Rrep } (\text{the } (\text{dhops } \text{(rt } \xi ) \text{(dip } \xi ))) \text{(dip } \xi )) \text{(sqn } (\text{rt } \xi ) \text{(dip } \xi )) \text{(oip } \xi )) \text{(ip } \xi ))) \) .

from ~\( \text{dip } \xi \in \text{vD (rt } \xi ) \)

have "dip \( \xi \in \text{vD (rt } \xi ) \)" by (rule vD_iD_gives_kD(1))

thus "0 < \( \text{the } (\text{dhops } \text{(rt } \xi ) \text{(dip } \xi )))" by simp

qed

lemma hop_count_zero_oip_dip_sip:
"\( \text{paodv } i =\rightarrow \text{ (recvmsg msg_zhops } \rightarrow \text{ onl } \Gamma_{\text{AODV }} (\lambda (\xi , l). (l \in \{\text{PAodv-:4..PAodv-:5} ) \cup \{PRreq-:n|n. True\} ) \rightarrow (\text{hops } \xi = 0 \rightarrow \text{oip } \xi = \text{sip } \xi ))) \) .

by (inv cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) auto

lemma osn_rreq:
"\( \text{paodv } i =\rightarrow \text{ (recvmsg rreq_rrep_sn } \rightarrow \text{ onl } \Gamma_{\text{AODV }} (\lambda (\xi , l). (l \in \{\text{PAodv-:4, PAodv-:5} ) \cup \{PRreq-:n|n. True\} ) \rightarrow 1 \leq \text{osn } \xi ))) \) .

by (inv cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma osn_rreq':
"\( \text{paodv } i =\rightarrow \text{ (recvmsg } (\lambda m. \text{rreq_rrep_sn m } \land \text{msg_zhops m}) \rightarrow \text{ onl } \Gamma_{\text{AODV }} (\lambda (\xi , l). (l \in \{\text{PAodv-:4, PAodv-:5} ) \cup \{PRreq-:n|n. True\} ) \rightarrow 1 \leq \text{osn } \xi ))) \) .

proof (rule invariant_weakenE [OF osn_rreq])

fix a
assume "recvmsg (\lambda m. \text{rreq_rrep_sn m } \land \text{msg_zhops m}) a"
thus "recvmsg rreq_rrep_sn a"
by (cases a) simp_all

qed

lemma dsn_rrep:
"\( \text{paodv } i =\rightarrow \text{ (recvmsg } \text{rreq_rrep_sn } \rightarrow \text{ onl } \Gamma_{\text{AODV }} (\lambda (\xi , l). (l \in \{\text{PAodv-:6, PAodv-:7} ) \cup \{PRrep-:n|n. True\} ) \rightarrow 1 \leq \text{dsn } \xi ))) \) .

by (inv cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma dsn_rrep':
"\( \text{paodv } i =\rightarrow \text{ (recvmsg } (\lambda m. \text{rreq_rrep_sn m } \land \text{msg_zhops m}) \rightarrow \text{ onl } \Gamma_{\text{AODV }} (\lambda (\xi , l). (l \in \{\text{PAodv-:6, PAodv-:7} ) \cup \{PRrep-:n|n. True\} ) \rightarrow 1 \leq \text{dsn } \xi ))) \) .

proof (rule invariant_weakenE [OF dsn_rrep])

fix a
assume "recvmsg (\lambda m. \text{rreq_rrep_sn m } \land \text{msg_zhops m}) a"
thus "recvmsg rreq_rrep_sn a"
by (cases a) simp_all
qed

lemma hop_count_zero_oip_dip_sip':
  "paodv i ||| (recvmsg (λ m. rreq_rrep_sn m ∧ msg_zhops m) |→ onl Γ_AODV (λ(ξ, _).
    (hops ξ = 0 −→ oip ξ = sip ξ))
    ∧ ((λ(ξ, _) |→ (hops ξ = 0 −→ dip ξ = sip ξ))))"
proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])
  fix a
  assume "recvmsg (λ m. rreq_rrep_sn m ∧ msg_zhops m) a"
  thus "recvmsg msg_zhops a" by (cases a) simp_all
qed

Proposition 7.12

lemma zero_seq_unk_hops_one':
  "paodv i ||| (recvmsg (λ m. rreq_rrep_sn m ∧ msg_zhops m) |→ onl Γ_AODV (λ(ξ, _).
    ∀ dip ∈ kD(rt ξ). (sqn (rt ξ) dip = 0 −→ sqnf (rt ξ) dip = unk)
    ∧ (sqnf (rt ξ) dip = unk −→ the (dhops (rt ξ) dip) = 1)
    ∧ (the (dhops (rt ξ) dip) = 1 −→ the (nhop (rt ξ) dip) = dip))"
proof -
  { fix dip and ξ :: 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 ξ).
      (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 ξ)"
    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 ξ).
      (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 ξ)"
    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 ξ).
      (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 ξ)"
    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_ok3 [simp] = this
(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))"
}

note fromsip [simp] = this

have prreq_ok5 [simp]: "∀ dip ∈ kD rt.
  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)

have prreq_ok6 [simp]: "∀ 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

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined]
onl_invariant_sterms [OF aodv_wf hop_count_zero_oip_dip_sip']
seq_step_invariant_sterms_TT [OF sqns_increase aodv_wf aodv_trans]
onl_invariant_sterms [OF aodv_wf osn_rreq']
onl_invariant_sterms [OF aodv_wf dsn_rrep'] clarsimp+)
qed

lemma zero_seq_unk_hops_one:
"paodv i |= (∀ 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))"
by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

lemma kD_unk_or_atleast_one:
"paodv i |= (∀ dip ∈ kD rt. π₃ (the (rt dip)) = unk ∨
  1 ≤ π₂ (the (rt dip)))"
proof -
{ fix sip rt dsn1 dsn2 ds1 ds2 flag1 flag2 hops1 hops2 nip1 nip2 pre1 pre2
assume "dsk1 = unk ∨ Suc 0 ≤ dsn2"
hence "π₃ (the (update rt sip (dsn1, dsk1, flag1, hops1, nip1, pre1)) sip)) = unk ∨
  Suc 0 ≤ sqn (update rt sip (dsn2, dsk2, flag2, hops2, nip2, pre2)) sip"
unfolding update_def by (cases "dsk1 = unk") (clarsimp split: option.split)+
} note fromsip [simp] = this

{ fix dip sip rt dsn1 dsn2 ds1 ds2 flag1 flag2 hops1 hops2 nip1 nip2 pre1 pre2
assume alldk: "∀ 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, nip1, pre1) dip)) = unk"
\[ \forall \text{Suc } 0 \leq \text{sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2)) dip} \]
\[
\text{(is } \forall \text{dip}\in\text{kD(rt). } \text{?prop dip)}
\]
proof
\[\text{fix dip}\]
assume "dip\in\text{kD(rt)}"
thus "?prop dip"
proof (cases "dip = sip")
assume "dip = sip"
with \(*\) show \(?thesis\)
by simp
next
assume "dip \neq sip"
with \(\langle\text{dip}\in\text{kD(rt)}\rangle\) allkd show \(?thesis\)
by simp
qed

\{ \text{note solve_update [simp] = this} \}

\{ \text{fix rt dests}\}
assume \(\ast\): "\(\forall \text{ip}\in\text{dom(dests). ip}\in\text{kD(rt)} \land \text{sqn rt ip} \leq \text{the (dests ip)}\)"
and \(\ast\ast\): "\(\forall \text{ip}\in\text{kD(rt). } \pi_3(\text{the (rt ip)}) = \text{unk} \lor \text{Suc } 0 \leq \text{sqn rt ip}\)"
have "\(\forall \text{dip}\in\text{kD(rt). } \pi_3(\text{the (rt dip)}) = \text{unk} \lor \text{Suc } 0 \leq \text{sqn (invalidate rt dests) dip}\)"
proof
\[\text{fix dip}\]
assume "dip\in\text{kD(rt)}"
with \(\ast\ast\) have "\(\pi_3(\text{the (rt dip)}) = \text{unk} \lor \text{Suc } 0 \leq \text{sqn rt dip}\)" ..
thus "\(\pi_3(\text{the (rt dip)}) = \text{unk} \lor \text{Suc } 0 \leq \text{sqn (invalidate rt dests) dip}\)"
proof
assume "\(\pi_3(\text{the (rt dip)}) = \text{unk}\)" thus \(?thesis\) ..
next
assume "\(\text{Suc } 0 \leq \text{sqn rt dip}\)"
have "\(\text{Suc } 0 \leq \text{sqn (invalidate rt dests) dip}\)"
proof (cases "dip\in\text{dom(dests)}")
assume "dip\in\text{dom(dests)}"
with \(\ast\) have "\(\text{sqn rt dip} \leq \text{the (dests dip)}\)" by simp
with \(\langle\text{Suc } 0 \leq \text{sqn rt dip}\rangle\) have "\(\text{Suc } 0 \leq \text{the (dests dip)}\)" by simp
with \(\langle\text{dip}\in\text{dom(dests)}\rangle\) \(\langle\text{dip}\in\text{kD(rt)}\rangle\) \(\text{[THEN kD_Some]}\) show \(?thesis\)
unfolding invalidate_def sqn_def by auto
next
assume "dip\in\text{dom(dests)}"
with \(\langle\text{Suc } 0 \leq \text{sqn rt dip}\rangle\) \(\langle\text{dip}\in\text{kD(rt)}\rangle\) \(\text{[THEN kD_Some]}\) show \(?thesis\)
unfolding invalidate_def sqn_def by auto
qed
thus \(?thesis\) by (rule disjI2)
qed
\{ \text{note solve_invalidate [simp] = this} \}

show \(?thesis\)
by (inv_c terms 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:
"\text{paodv } i \mid=_{A} (\text{recvmsg } \text{rreq}_rrep_{-s} \rightarrow \text{onll } \Gamma_{AODV}(\lambda(\_ , a, _). \text{anycast } \text{rreq}_rrep_{-s} a) )"
proof -
have sqnf_kno: "\text{paodv } i \mid=_{A} \text{onll } \Gamma_{AODV}(\lambda(\xi, 1). \text{\_})",
\(\langle 1 \in \{\text{PRreq:-16..PRreq:-18} \rightarrow \text{sqnf (rt } \xi) (\text{dip } \xi) = \text{kno}\}\rangle\)"
by (inv_c terms inv add: onl_invariant_sterms TT [OF aodv_wf addpreRT_welldefined])
\}

Proposition 7.14

Lemma rreq_rrep_fresh_any_step_invariant:
"paodv i \vdash_G onl \Gamma_{AODV} (\lambda((\xi, _), a, _). anycast (rreq_rrep_fresh (rt \xi)) a)"

Proof

Have rreq_oip: "paodv i \vdash onl \Gamma_{AODV} (\lambda(\xi, l). anycast (rreq_oip \xi) a)"

Proof inv_cterms

Fix l l' pp p'
Assume "((\xi, pp) \in reachable (paodv i) TT"
And "\{PRreq-2\} [\xi. \xi[rt := update (rt \xi) (\xi) (osn \xi, kno, val, Suc \{hops \xi, sip \xi, {}\})] p' \in sterms \Gamma_{AODV} pp" And "l' = PRreq-3"

Show "osn \xi < sqn (update (rt \xi) (\xi) (osn \xi, kno, val, Suc \{hops \xi, sip \xi, {}\})) (osn \xi) \lor (sqn (update (rt \xi) (\xi) (osn \xi, kno, val, Suc \{hops \xi, sip \xi, {}\})) (osn \xi)) (osn \xi) \leq Suc \{hops \xi, sip \xi, {}\}) (osn \xi) \land the (flag (rt \xi) (\xi) (osn \xi, kno, val, Suc \{hops \xi, sip \xi, {}\})) (osn \xi) = val)"

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

Qed

Have rrep_prrep: "paodv i \vdash onl \Gamma_{AODV} (\lambda(\xi, l). anycast (rrep_prrep \xi) a)"

Proof

By (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_inv])

Qed

Proposition 7.15

Lemma rrerr_invalid_any_step_invariant:
"paodv i \vdash_G onl \Gamma_{AODV} (\lambda((\xi, _), a, _). anycast (rrerr_invalid (rt \xi)) a)"

Proof -

Have dests_inv: "paodv i \vdash_G onl \Gamma_{AODV} (\lambda(\xi, l). (l \in \{PAodv-15, PPkt-7, PRreq-9, PRreq-21, PRreq-10, PRerr-1\} --> (\forall ip \in dom(dests \xi). ip \in vD(rt \xi)) \land (l \in \{PAodv-16..PAodv-19\} \lor \{PPkt-8..PPkt-11\} \lor \{PRreq-10..PRreq-13\}))

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 \(ip \mapsto sqn\), 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 \(\text{dests\_vd\_inc\_sqn}\):

\[
\text{paodv } i \mid=
\text{onl } \Gamma_{AODV} (\lambda(\xi, 1). (1 \in \{PAodv\text{-}15, PPkt\text{-}7, PRreq\text{-}9, PRreq\text{-}21, PRrep\text{-}10\})
\rightarrow (∀ip\in\text{dom}(dests }\xi). ip\in\text{vD}(rt }\xi \land \text{the } (dests }\xi ip) = \text{inc } (sqn (rt }\xi ip))
\land (l = PPkt\text{-}14 \rightarrow dip }\xi \in\text{vD}(rt }\xi))
\]

by \(\text{inv\_cterms (clarsimp split: split_if_asm option.split_asm simp: domIff)+}\)

Proposition 7.27

lemma \(\text{route\_tables\_fresher}\):

\[
\text{paodv } i \mid_A (recvmsg rreq_rrep_sn \rightarrow \text{onl } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall dip \in kD(rt }\xi). rt }\xi \subseteq dip rt }\xi')
\]

proof \(\text{inv\_cterms inv add: onl\_invariant\_sterms [OF aodv_wf dests_inv]}\)

fix \(\xi pp p'\)
assume \(\((\xi, pp) \in \text{reachable (paodv } i) (recvmsg rreq_rrep_sn)\)\)
and \(\{\text{PRreq\text{-}2}\}\{\xi. \xi(rt := \text{update } (rt }\xi) (oip }\xi) (\text{osn }\xi, \text{kno, val, Suc } (\text{hops }\xi), \text{sip }\xi, \{\})\}\)
\(p' \in \text{sterms } \Gamma_{AODV} pp\)
and \("\text{Suc } 0 \leq \text{osn }\xi"
and *: \("\forall ip \in kD (rt }\xi). \text{Suc } 0 \leq \text{the } (\text{dhops } (rt }\xi ip)"
show \("\forall ip \in kD (rt }\xi). \text{rt }\xi \subseteq dip \text{update } (rt }\xi) (\text{oip }\xi) (\text{osn }\xi, \text{kno, val, Suc } (\text{hops }\xi), \text{sip }\xi, \{})\)"
proof
fix ip
assume "ip \in kD (rt }\xi)"
moreover with * have "1 \leq \text{the } (\text{dhops } (rt }\xi ip)" by simp
moreover from Suc 0 \leq \text{osn }\xi:
have "\text{update\_arg\_wf } (\text{osn }\xi, \text{kno, val, Suc } (\text{hops }\xi), \text{sip }\xi, \{})" ..
ultimately show "\text{rt }\xi \subseteq dip \text{update } (rt }\xi) (\text{oip }\xi) (\text{osn }\xi, \text{kno, val, Suc } (\text{hops }\xi), \text{sip }\xi, \{})" by (rule rt_fresher_update)
qed
∀ ip ∈ kD (rt ξ). rt ξ ⊑ ip update (rt 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) (dsn ξ, kno, val, Suc (hops ξ), sip ξ, {})" by (rule rt_fresher_update)
qued

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))⟩

54
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 ∨ (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)

55
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 "sqn (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)
assumes "ip ∈ D(σ, sip)"
hence "the (flag (σ, sip)) ip = inv" ..
with ⟨sn = nsqn (σ, sip)⟩ show ?thesis
  by simp

next

assumes "ip ∈ D(σ, sip)"
hence "nsqn (σ, sip) ip = sqn (σ, sip) ip" ..
with ⟨sn = nsqn (σ, sip)⟩ show "ip ∈ D (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 (σ, sip) ∧ sn ≤ nsqn (σ, sip) ip
∧ (nsqn (σ, sip) ip = sn
  → (the (dhops (σ, sip)) ip) ≤ hops
  ∨ the (flag (σ, sip) ip) = inv))"
shows "ip ∈ kD (σ, sip) ∧ sn ≤ nsqn (σ, sip) ip
∧ (nsqn (σ, sip) ip = sn
  → (the (dhops (σ, sip)) ip) ≤ hops
  ∨ the (flag (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 clar simp (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 ∈ D (σ, sipc) ∧ nsqn (σ, sipc) oipc ≥ osnc
∧ (nsqn (σ, sipc) oipc = osnc
  → (hopsc ≥ the (dhops (σ, sipc)) oipc)
  ∨ the (flag (σ, sipc) oipc) = inv))
| Rrep hopsc dipc dsnc _ sipc ⇒ dsnc ≥ 1 ∧ (sipc ≠ dipc → dipc ∈ D (σ, sipc) ∧ nsqn (σ, sipc) dipc ≥ dsnc
∧ (nsqn (σ, sipc) dipc = dsnc
  → (hopsc ≥ the (dhops (σ, sipc)) dipc)
  ∨ the (flag (σ, sipc) dipc) = inv))
| Rerr destsc sipc ⇒ (∀ ripc ∈ dom(destsc). (ripc ∈ D (σ, sipc))
∧ the (destsc ripc) - 1 ≤ nsqn (σ, sipc) ripc)
| _ ⇒ True"
lemma msg_fresh [simp]:
"\( \land \text{ hops rreqid dip dsn dsk oip osn sip.} \)
\[
\text{msg\_fresh} \ \sigma \ \text{(Rreq hops rreqid dip dsn dsk oip osn sip)} =
\begin{align*}
\text{(osn} & \geq 1 \land (\text{sip} \neq \text{oip} \rightarrow \text{oip} \in \text{D}(\text{rt}(\sigma \text{ sip}))) \\
& \land \text{nsqn (rt}(\sigma \text{ sip})) \text{ oip} \geq \text{osn} \\
& \land (\text{nsqn (rt}(\sigma \text{ sip})) \text{ oip} = \text{osn} \\
& \rightarrow (\text{hops} \geq \text{the (dhops (rt}(\sigma \text{ sip})) \text{ oip)}} \\
& \lor \text{the (flag (rt}(\sigma \text{ sip})) \text{ oip) = inv))))")
\end{align*}
"\\
"\( \land \text{ hops dip dsn oip sip.} \text{msg\_fresh} \ \sigma \ \text{(Rrep hops dip dsn oip sip)} =
\begin{align*}
\text{(dsn} & \geq 1 \land (\text{sip} \neq \text{dip} \rightarrow \text{dip} \in \text{D}(\text{rt}(\sigma \text{ sip}))) \\
& \land \text{nsqn (rt}(\sigma \text{ sip})) \text{ dip} \geq \text{dsn} \\
& \land (\text{nsqn (rt}(\sigma \text{ sip})) \text{ dip} = \text{dsn} \\
& \rightarrow (\text{hops} \geq \text{the (dhops (rt}(\sigma \text{ sip})) \text{ dip)}} \\
& \lor \text{the (flag (rt}(\sigma \text{ sip})) \text{ dip) = inv))))")
\end{align*}
"\\
"\( \land \text{ dests sip.} \text{msg\_fresh} \ \sigma \ \text{(Rerr dests sip)} =
\begin{align*}
\forall \text{ripc} \in \text{dom(dests)}. (\text{ripc} \in \text{D}(\text{rt}(\sigma \text{ sip}))) \\
& \land \text{the (dests ripc) - 1} \leq \text{nsqn (rt}(\sigma \text{ sip})) \text{ ripc})
\end{align*}
"\\
"\( \land \text{ d dip.} \text{msg\_fresh} \ \sigma \ \text{(Newpkt d dip)} = \text{True}"
\"\( \land \text{ d dip sip.} \text{msg\_fresh} \ \sigma \ \text{(Pkt d dip sip)} = \text{True}"
unfolding msg\_fresh\_def by simp_all

lemma msg\_fresh\_inc\_sn [simp, elim]:
"msg\_fresh \ \sigma \ m =\Rightarrow \text{rreq\_rrep\_sn m}"
by (cases m) simp_all

lemma recv\_msg\_fresh\_inc\_sn [simp, elim]:
"orecvmsg (msg\_fresh) \ \sigma \ m =\Rightarrow \text{recvmsg rreq\_rrep\_sn m}"
by (cases m) simp_all

lemma rreq\_nsqn\_is\_fresh [simp]:
fixes \( \sigma \ \text{msg hops rreqid dip dsn dsk oip osn sip} \)
assumes "\text{rreq\_rrep\_fresh (rt}(\sigma \text{ sip})) (\text{Rreq hops rreqid dip dsn dsk oip osn sip})" 
and "\text{rreq\_rep\_sn (\text{Rreq hops rreqid dip dsn dsk oip osn sip})"
shows "\text{msg\_fresh} \ \sigma \ \text{(Rreq hops rreqid dip dsn dsk oip osn sip)}"
(is "\text{msg\_fresh} \ \sigma \ \text{msg}\)"
proof -
  let ?rt = "rt (\sigma \ \text{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 "\text{sip} \neq \text{oip}"
  with assms(1) show "\text{oip} \in \text{D}(\text{rt})" by simp
next
  assume "\text{sip} \neq \text{oip}"
  and \"\text{nsqn ?rt oip = osn}\"
  show "\text{the (dhops ?rt oip) \leq hops \lor the (flag ?rt oip) = inv}"
proof (cases "\text{oip} \in \text{vD}(\text{rt})")
  assume "\text{oip} \in \text{vD}(\text{rt})"
  hence "\text{nsqn ?rt oip = sqn ?rt oip} "..
  with (\text{nsqn ?rt oip = osn}) have "\text{sqn ?rt oip = osn}" by simp
  with assms(1) and (\text{sip} \neq \text{oip}) have "\text{the (dhops ?rt oip) \leq hops}"
    by simp 
  thus ?thesis ..
next
  assume "\text{oip} \in \text{vD}(\text{rt})"
  moreover from assms(1) and (\text{sip} \neq \text{oip}) have "\text{oip} \in \text{D}(\text{rt})" by simp
  ultimately have "\text{osnip ?rt oip} " by auto
  hence "\text{the (flag ?rt oip) = inv} "..
  thus ?thesis ..
  qed
next
  assume "\text{sip} \neq \text{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)
  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 "osn ≤ nsqn ?rt oip" by simp
thus "osn ≤ nsqn ?rt oip" by simp
qed
qed simp

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"
    byclarsimp
  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
  have * 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))⟩ have "rip ∈ kD(rt (σ sip)) ∧ the (dests rip) - 1 ≤ nsqn (rt (σ sip)) rip"
  by clarsimp
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 oip osn sip
  assume [simp]: "m = Rreq hops rreqid dip dsn oip osn sip"
  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)))"
  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)))"
  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

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 σ AODV’ :: "((ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp)) set"
where "σ AODV’ ≡ {λ i. aodv_init i, Γ AODV PAodv}"
abbreviation opaodv :: "ip ⇒ ((ip ⇒ state) × (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
where "opaodv i ≡ |init = σ AODV’, trans = oseqp_sos Γ AODV i|
lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))"
unfolding σ AODV’_def σ AODV’_labels by rule simp_all
lemma oaodv_control_within [simp]: "control_within Γ AODV (init (opaodv i))"
unfolding σ AODV’_def by (rule control_withinI) (auto simp del: Γ AODV_simp)
lemma σ AODV’_labels [simp]: "(σ, p) ∈ σ AODV’ ⟹ labels Γ AODV p = {PAodv-:0}"
unfolding σ AODV’_def by simp
lemma oaodv_init_kD_empty [simp]: "(σ, p) ∈ σ AODV’ ⟹ kD (rt (σ i)) = {}"
unfolding σ AODV’_def kD_def by simp
lemma oaodv_init_vD_empty [simp]: "(σ, p) ∈ σ AODV’ ⟹ vD (rt (σ i)) = {}"
unfolding σ AODV’_def vD_def by simp
lemma oaodv_trans: "trans (opaodv i) = oseqp_sos Γ 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

0.10 Global invariant proofs over sequential processes

theory Global_Invariants
imports Seq_Invariants
       Aodv_Predicates
       Fresher

lemma other_quality_increases [elim]:
assumes "other quality_increases I σ σ'"
shows "∀ j. quality_increases (σ j) (σ' j)"
using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
fixes m
assumes *: "otherwith P I (orevmsg Q) σ σ' a"
and weakenP: "∀ σ m. P σ m =⇒ P' σ m"
and weakenQ: "∀ σ m. Q σ m =⇒ Q' σ m"
shows "otherwith P' I (orevmsg Q') σ σ' a"
proof
fix j
assume "j /∈ I" with * have "P (σ j) (σ' j)" by auto
thus "P' (σ j) (σ' j)" by (rule weakenP)
next
from * have "orevmsg Q σ a" by auto
thus "orevmsg Q' σ a" by rule (erule weakenQ)
qed

lemma orceived_msg_inv:
assumes other: "∀ σ σ' m. [ [ P σ m; other Q {i} σ σ' ] ] =⇒ P σ' m"
and local: "∀ σ m. P σ m =⇒ P (σ (i := σ i (| msg := m |))) m"
shows "opaodv i ||= (otherwith Q {i} (orevmsg P), other Q {i} →) onll Γ_AODV (λ(σ, l). l ∈ {PAodv-:1} → P σ (msg (σ i)))"
proof (inv_cterms, intro impI)
fix σ σ' l
assume "l = PAodv-:1 → P σ (msg (σ i))"
and "l = PAodv-:1"
and "other Q {i} σ σ'"
from this(1-2) have "P σ (msg (σ i)) .." .
hence "P σ' (msg (σ i))" using "other Q {i} σ σ'"
by (rule other)
moreover from "other Q {i} σ σ'" have "σ' i = σ i .."
ultimately show "P σ' (msg (σ' i))" by simp
next
fix σ σ' msg
assume "otherwith Q {i} (orevmsg 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:
"paodv i |=A (revcmsg req_rrep_sn → onll Γ_AODV (λ((ξ, _), _, (ξ', _)). quality_increases ξ ξ'))"
proof (rule step_invariantI)
fix s a s'
assume sr: "s ∈ reachable (paodv i) (revcmsg req_rrep_sn)"
and tr: "(s, a, s') ∈ trans (paodv i)"
and rm: "recvmsg rreq_rrep_sn a"
from sr have srTT: "s ∈ reachable (paodv i) TT" ..
from route_tables_fresher sr tr rm
have "onll Γ AODV (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 Γ AODV (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 Γ AODV (λ((ξ, _), _, (ξ', _)). ∀ ip. sqn (rt ξ) ip ≤ sqn (rt ξ') ip) (s, a, s')"
by (rule step_invariantD)
ultimately show "onll Γ AODV (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_aodv oaodv_trans aodv_trans,
simplified seqll_onll_swap]

lemma quality_increases:
"opaodv i \= A (otherwith quality_increases {i} (orecvmsg (λ_. rreq_rrep_sn)),
other quality_increases {i} →)
onll Γ AODV (λ((σ, _), _, (σ', _)). ∀ j. quality_increases (σ j) (σ' j))"(is "\= A (?S, _ → _)")proof (rule onll_ostep_invariantI, simp)
fix σ p a σ' p' l'
assume or: "((σ, p), a, (σ', p')) ∈ oseqp_sos Γ AODV i"
assume ll: "l ∈ labels Γ AODV p"
and ll': "l' ∈ labels Γ AODV 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), a, (σ', p')) ∈ oseqp_sos Γ AODV i"
by (drule onll_ostep_invariantD [OF olocal_quality_increases], auto simp: seqll_def)
with ⟨?S σ σ' a⟩ have "∀ j. quality_increases (σ j) (σ' j)"by (auto dest!: otherwith_syncD)
qed

lemma rreq_rrep_nsqn_fresh_any_step_invariant:
"opaodv i \= A (act (recvmsg rreq_rrep_sn), other A {i} →)
onll Γ AODV (λ((σ, _), a, _). anycast (msg_fresh σ) a)"
proof (rule ostep_invariantI, simp del: act_simp)
fix σ p a σ' p' l'
assume or: "((σ, p), a, (σ', p')) ∈ oseqp_sos Γ AODV i"
and recv: "act (recvmsg rreq_rrep_sn) σ a"
obtain l l' where "l ∈ labels Γ AODV p" and "l' ∈ labels Γ AODV p'"by (metis aodv_ex_label)
from ⟨((σ, p), a, (σ', p'))⟩ have tr: "((σ, p), a, (σ', p')) ∈ trans (opaodv i)" by simp
have "anycast (rreq_rrep_fresh (rt (σ i))) a"proof -
have "opaodv i \= A (act (recvmsg rreq_rrep_sn), other A {i} →)"
onll Γ_AODV (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_aodv,
simplified seqll_onll_swap]]) auto
hence "onll Γ_AODV (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 Γ_AODV p⟩ and ⟨l'∈ labels Γ_AODV p'⟩ by auto
qed

moreover have "anycast (rerr_invalid (rt (σ i))) a"
proof -
have "opaodv i |=A (act (recvmsg rreq_rrep_sn), other A {i} →)
onll Γ_AODV (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_aodv,
simplified seqll_onll_swap]]) auto
hence "onll Γ_AODV (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 Γ_AODV p⟩ and ⟨l'∈ labels Γ_AODV p'⟩ by auto
qed

moreover have "anycast rreq_rrep_sn a"
proof -
from or tr recv
have "onll Γ_AODV (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_aodv
oaoav_trans aodv_trans,
simplified seqll_onll_swap]])
thus ?thesis
using ⟨l∈ labels Γ_AODV p⟩ and ⟨l'∈ labels Γ_AODV p'⟩ by auto
qed

moreover have "anycast (λm. not_Pkt m −→ msg_sender m = i) a"
proof -
have "opaodv i |=A (act (recvmsg rreq_rrep_sn), other A {i} →)
onll Γ_AODV (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
hence "onll Γ_AODV (seqll i (λ((ξ, _), a, _). anycast (λm. not_Pkt m −→ msg_sender m = i) a))
((σ, p), a, (σ', p'))"
using or tr recv by - (drule(3) onll_ostep_invariantD, auto)
thus ?thesis using ⟨l∈ labels Γ_AODV p⟩ and ⟨l'∈ labels Γ_AODV p'⟩ by 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 Γ_AODV (λ((σ, _), a, _). anycast (msg_fresh σ) a) ((σ, p), a, (σ', p'))"
by auto
qed

lemma oreceived_rreq_rrep_msqn_fresh_inv:
"opaodv i |= (otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} →)
onll Γ_AODV (λ(σ, 1). l ∈ {PAodv−:1} → msg_fresh σ (msg (σ i)))"
proof (rule oreceived_msg_inv)
fix σ σ'
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 quality_increases_nsqn_fresh:
"opaoav i |=A (otherwith quality_increases {i} (orecvmmsg msg_fresh),
other quality_increases {i} →)
onl Γ AODV (λ(⟨σ, _, _⟩, ⟨σ', _⟩). ∀ j. quality_increases (σ j) (σ' j))"
by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
"opaoav i |= (otherwith quality_increases {i} (orecvmmsg msg_fresh),
other quality_increases {i} →)
onl Γ AODV (seql (λ(⟨l, i⟩. l ∈ {PAodv:-4, PAodv:-5} ∪ {PRreq:-n |n. True} → 1 ≤ osn ⟨⟩))"
by (rule oinvariant_weakenE [OF open_seq_invariant [OF oosn_rreq_initial_aodv]])
(auto simp: seql_onl_swap)

lemma rreq_sip:
"opaoav i |= (otherwith quality_increases {i} (orecvmmsg msg_fresh),
other quality_increases {i} →)
onl Γ AODV (λ(⟨σ, 1⟩).
(1 ∈ {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 aodvTrans]
oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
oinvariant_sterms [OF aodv_wf oosn_rreq]
simp add: seqsimpl
simp del: One_nat_def, rule impI)
fix σ σ' p l
assumes "(σ, p) ∈ oreachable (opaoav i) ?S ?U"
and "1 ∈ labels Γ AODV p" and pre:
"(1 = PAodv:-4 ∨ 1 = PAodv:-5 ∨ 1 = PRreq:-0 ∨ 1 = 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)"
and "other quality_increases {i} σ σ'" and hyp: "(1 = PAodv:-4 ∨ 1 = PAodv:-5 ∨ 1 = PRreq:-0 ∨ 1 = 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))))
∧ nsqn (σ' i) ≤ nsqn (rt (σ' (sip (σ' i)))) (oip (σ' i))
∧ (nsqn (rt (σ' (sip (σ' i)))) (oip (σ' i)) = osn (σ' i))
→ (hops (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: (simp (σ i) ≠ i))
moreover from ⟨(σ, p) ∈ ooreachable (opaodv i) ⟩ ⟨S ⟩ ⟨U ⟩ {l ∈ labels Γ_{AODV} p} and hyp
have "1 ≤ osn (σ′ i)"
  by (auto dest!: onl_oinvariant_weakenD [OF oosn_rreq]
      simp add: seqsimplp (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

lemma odsn_rrep:
"opaodv i = (otherwith quality_increases {i} {orecvmsg msg_fresh},
             other_quality_increases {i} →)
onl Γ_{AODV} (seql (λ(i, l). l ∈ {PAodv-:6, PAodv-:7} ∪ {PRrep-:n|n. True} → 1 ≤ dsn (ξ))"
by (rule oinvariant_weakenE [OF open_seq_invariant [OF dsn_rrep initiali_aodv]])
(auto simp: seql_onl_swap)

lemma rrep_sip:
"opaodv i = (otherwith quality_increases {i} {orecvmsg msg_fresh},
             other_quality_increases {i} →)
onl Γ_{AODV} (λ(σ, i).
  (λ (l ∈ {PAodv-:6, PAodv-: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 quality_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 odsn_rrep]
  simp del: One_nat_def, rule impl)
fix σ σ′ ∈ Γ
assume "(σ, p) ∈ ooreachable (opaodv i) ⟩ ⟨S ⟩ ⟨U ⟩
and "l ∈ labels Γ_{AODV} p"
and pre:
"((1 = PAodv-:6 ∨ l = PAodv-:7 ∨ l = PRrep-:0 ∨ l = PRrep-:1) ∧ sip (σ i) ≠ dip (σ i))
  → 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))
  → (hops (σ i) ≥ the (dhops (rt (σ (sip (σ i)))) (dip (σ i))))
  ∨ the (flag (rt (σ (sip (σ i)))) (dip (σ i))) = inv)"
and "other_quality_increases {i} σ σ′"
and hyp: "((1 = PAodv-:6 ∨ l = PAodv-:7 ∨ l = PRrep-:0 ∨ l = PRrep-:1) ∧ sip (σ′ i) ≠ dip (σ′ i))
  (is "labels ∧ sip (σ′ i) ≠ dip (σ′ i)")
from this(4) have "σ′ i = σ i" ..
with hyp have hyp': "?labels ∧ sip (σ i) ≠ dip (σ i)" by simp
show "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)"
lemma rerr_sip:  
"opaodv i |- (otherwith quality_increases \{i\} (orecvmsg msg_fresh),
  other quality_increases \{i\}) \rightarrow
  onl \Gamma_{AODV} (\lambda(\sigma, i).
    \forall i \in \{PAodv-:8, PAodv-:9, PRerr-:0, PRerr-:1\}
    \rightarrow (\forall ripc\in dom(dests (\sigma i)). ripc\in kd(rt (\sigma (sip (\sigma i))))) \land
    the (dests (\sigma i) ripc) - 1 \leq nsqn (rt (\sigma (sip (\sigma i)))) ripc))"

(is "_ | (?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))
    \land 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 \in kd(rt (\sigma sip)) \land rsn - 1 \leq nsqn (rt (\sigma sip)) rip"
proof
  from (rip\in kd(rt (\sigma sip))) and (quality_increases (\sigma sip) (\sigma sip))
  show "rip \in kd(rt (\sigma sip))" ..
next
  from (rip\in kd(rt (\sigma sip))) and (quality_increases (\sigma sip) (\sigma sip))
  have "nsqn (rt (\sigma sip)) rip \leq nsqn (rt (\sigma sip)) rip" ..
  with "rsn - 1 \leq nsqn (rt (\sigma sip)) rip" 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 equality_increases_msg_fresh aodv_wf
    aodv_trans]
    oseq_step_invariant_sterms [OF aodv_wf oreceived_rreq_rrep_msg_fresh_inv]
    other quality_increases other_localD
  simp del: One_nat_def, intro conj1)
lemma prerr_guard: "paodv i |=
  onl Γ_AODV (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_aodv oadv_trans aodv_trans,
    simplified seql_onl_swap,
    THEN oinvariant_anyact]

lemmas odests_vD_inc_sqn =
  open_seq_invariant [OF dests_vD_inc_sqn initiali_aodv oadv_trans aodv_trans,
    simplified seql_onl_swap,
    THEN oinvariant_anyact]

lemmas oprerr_guard =
  open_seq_invariant [OF prerr_guard initiali_aodv oadv_trans aodv_trans,
    simplified seql_onl_swap,
    THEN oinvariant_anyact]

Proposition 7.28

lemma seq_compare_next_hop':
  "opaodv i |= (otherwith quality_increases {i} (orecmsg msg_fresh),
  other quality_increases {i} → onl Γ_AODV (λ(σ, _).
    (∀ dip. let nhip = the (nhop (rt (σ i)) dip)
      in dip ∈ kD(rt (σ i)) ∧ nhip ≠ dip →
      dip ∈ kD(rt (σ nhip)) ∧ sqn (rt (σ i)) dip ≤ sqn (rt (σ nhip)) 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))) ∧ sqn (rt (σ i)) dip ≤ sqn (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))) ∧ sqn (rt (σ i)) dip ≤ sqn (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 "sqn (rt (σ i)) dip ≤ sqn (rt (σ (nhop dip))) dip"
    by auto
    from qinc have qinc_nh: "quality_increases (σ (nhop dip)) (σ' (nhop dip))" ..
    with "dip∈kD(rt (σ (nhop dip)))" have "dip∈kD (rt (σ' (nhop dip)))" ..
    moreover have "sqn (rt (σ i)) dip ≤ sqn (rt (σ' (nhop dip))) dip"
    proof -
      from "dip∈kD(rt (σ (nhop dip)))" qinc_nh
      have "sqn (rt (σ (nhop dip))) dip ≤ sqn (rt (σ' (nhop dip))) dip" ..
      with "sqn (rt (σ i)) dip ≤ sqn (rt (σ (nhop dip))) dip" show ?thesis
      by simp
    qed

    ultimately show "dip∈kD(rt (σ' (nhop dip)))
      ∧ sqn (rt (σ i)) dip ≤ sqn (rt (σ' (nhop dip))) dip" ..
  qed
}

note basic = this
\[
\text{Fix nhop and } \sigma' :: "ip \Rightarrow \text{state}"
\]
\[
\begin{align*}
\text{Assume pre: } & \forall \text{ dip } \in \text{kD}(rt (\sigma i)). \ \text{nhop dip } \neq \text{ dip } \rightarrow \text{ dip } \in \text{kD}(rt (\sigma (\text{nhop dip}))) \\
& \land \text{nsqn (rt (\sigma i)) dip } \leq \text{nsqn (rt (\sigma (\text{nhop dip}))) dip} \\
\land \text{ndest: } & \forall \text{ ripc } \in \text{dom (dests (\sigma i))}. \ \text{ripc } \in \text{kD (rt (\sigma (sip (\sigma i)))))} \\
& \land \ (\text{dests (\sigma i) ripc}) - 1 \leq \text{nsqn (rt (\sigma (sip (\sigma i)))) ripc} \\
\land \text{ issip: } & \forall \text{ ip } \in \text{dom (dests (\sigma i))}. \ \text{nhop ip } = \text{ sip (\sigma i)} \\
\land \text{ qinc: } & \forall \text{ j. quality_increases (\sigma j) (\sigma' j)}\]}

\[
\text{Have } \forall \text{ dip } \in \text{kD}(rt (\sigma i)). \ \text{nhop dip } \neq \text{ dip } \rightarrow \text{ dip } \in \text{kD (rt (\sigma' (nhop dip)))} \\
\land \text{nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip } \leq \text{nsqn (rt (\sigma' (nhop dip))) dip} \\
\text{Proof (intro ballI impI)}
\]
\[
\text{Fix dip}
\]
\[
\text{Assume } \text{"dip } \in \text{kD}(rt (\sigma i))"
\]
\[
\text{And } \text{"nhop dip } \neq \text{ dip"}
\]
\[
\text{With pre and qinc have } \text{"dip } \in \text{kD(rt (\sigma' (nhop dip)))"}
\]
\[
\text{And } \text{"nsqn (rt (\sigma i)) dip } \leq \text{nsqn (rt (\sigma' (nhop dip))) dip"}
\]
\[
\text{By (auto dest!: basic)}
\]
\[
\text{Have } \text{"nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip } \leq \text{nsqn (rt (\sigma' (nhop dip))) dip"}
\]
\[
\text{Proof (cases "dip } \in \text{dom (dests (\sigma i))")}
\]
\[
\text{Assume } \text{"dip } \in \text{dom (dests (\sigma i))"
\]
\[
\text{With } \text{dip } \in \text{kD}(rt (\sigma i)); \text{ obtain dsn where } \text{"dests (\sigma i) dip } = \text{ Some dsn"}
\]
\[
\text{By auto}
\]
\[
\text{With } \text{dip } \in \text{kD}(rt (\sigma i)); \text{ have } \text{"nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip } = \text{ dsn } - 1"
\]
\[
\text{By (rule nsqn_invalidate_eq)
\]
\[
\text{Moreover have } \text{"dsn } - 1 \leq \text{nsqn (rt (\sigma' (nhop dip))) dip"}
\]
\[
\text{Proof -
\]
\[
\text{From } \text{dests (\sigma i) dip } = \text{ Some dsn}; \text{ have } \text{"the (dests (\sigma i) dip ) } = \text{ dsn" by simp}
\]
\[
\text{With ndest and } \text{dip } \in \text{dom (dests (\sigma i)) have } \text{"dip } \in \text{kD (rt (\sigma (sip (\sigma i))))"}
\]
\[
\text{"dsn } - 1 \leq \text{nsqn (rt (\sigma (sip (\sigma i))) dip"}
\]
\[
\text{By auto
\]
\[
\text{Moreover from issip and } \text{dip } \in \text{dom (dests (\sigma i)) have } \text{"nhop dip } = \text{ sip (\sigma i)" ..
\]
\[
\text{Ultimately have } \text{"dip } \in \text{kD (rt (\sigma (nhop dip)))"
\]
\[
\text{And } \text{"dsn } - 1 \leq \text{nsqn (rt (\sigma (nhop dip))) dip" by auto}
\]
\[
\text{With qinc show } \text{"dsn } - 1 \leq \text{nsqn (rt (\sigma' (nhop dip))) dip"}
\]
\[
\text{By simp (metis kD_nsqn_quality_increases_trans)
\]
\[
\text{Qed
\]
\[
\text{Ultimately show } ?\text{thesis by simp
\]
\[
\text{Next assume } \text{"dip } \notin \text{ dom (dests (\sigma i))"
\]
\[
\text{With } \text{dip } \in \text{kD}(rt (\sigma i)); \text{ have } \text{"nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip } = \text{nsqn (rt (\sigma i)) dip"}
\]
\[
\text{By (rule nsqn_invalidate_other)
\]
\[
\text{With } \text{nsqn (rt (\sigma i)) dip } \leq \text{nsqn (rt (\sigma' (nhop dip))) dip}; \text{ show } ?\text{thesis by simp
\]
\[
\text{Qed
\]
\[
\text{With } \text{dip } \in \text{kD}(rt (\sigma' (nhop dip)))
\]
\[
\text{Show } \text{"dip } \in \text{kD (rt (\sigma' (nhop dip)))"}
\]
\[
\land \text{nsqn (invalidate (rt (\sigma i)) (dests (\sigma i))) dip } \leq \text{nsqn (rt (\sigma' (nhop dip))) dip" ..
\]
\[
\text{Qed
\]
\[
\text{Note } \text{basic_prerr } = \text{ this
\]
\[
\text{Fix } \sigma' :: "ip \Rightarrow \text{state"
\]
\[
\text{Assume a1: } \forall \text{ dip } \in \text{kD}(rt (\sigma i)). \ \text{the (nhop (rt (\sigma i)) dip) } \neq \text{ dip } \rightarrow \text{ dip } \in \text{kD(rt (\sigma (the (nhop (rt (\sigma i)) dip))))}
\]
\[
\land \text{nsqn (rt (\sigma i)) dip } \leq \text{nsqn (rt (\sigma (the (nhop (rt (\sigma i)) dip)))) dip"
\]
\[
\text{And a2: } \forall \text{ j. quality_increases (\sigma j) (\sigma' j)}
\]
\[
\text{Have } \forall \text{ dip } \in \text{kD}(rt (\sigma i)).
\]
\[
\text{The (nhop (update (rt (\sigma i)) (sip (\sigma i)) (0, unk, val, Suc 0, sip (\sigma i), {})) dip) } \neq \text{ dip } \rightarrow \text{ dip } \in \text{kD(rt (\sigma' (the (nhop (update (rt (\sigma i)) (sip (\sigma i)) (0, unk, val, Suc 0, sip (\sigma i), {})) dip)))}
\]
\[
\land \text{nsqn (update (rt (\sigma i)) (sip (\sigma i)) (0, unk, val, Suc 0, sip (\sigma i), {})) dip } \leq \text{nsqn (rt (\sigma' (the (nhop (update (rt (\sigma i)) (sip (\sigma i)) (0, unk, val, Suc 0, sip (\sigma i), {})) dip)))
\]
\[
\]
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 → oip ∈ kD (rt (σ oip)) (osn, kno, val, Suc hops, sip) oip ≠ oip → oip ∈ kD rt (σ' (the (nhop (rt (σ i)) dip))) ∧ nsqn (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 (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) oip" by (rule basic)

(is "?oip_in_kD ∧ ?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 ∧ ?oip_in_kD" by auto
next
assume rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"
and notoip: "?oip_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 ∧ ?oip_in_kD" ..
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)


Proposition 7.30

lemmas okD_unk_or_atleast_one =
  open_seq_invariant [OF kD_unk_or_atleast_one initiali_aodv,
  simplified seql_onl_swap]
lemmas ozero_seq_unk_hops_one =
    open_seq_invariant [OF zero_seq_unk_hops_one initiali_aodv,
                        simplified seql_onl_swap]

lemma oreachable_fresh_okD_unk_or_atleast_one:
  fixes dip
  assumes "(σ, p) ∈ oreachable (opaodv 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 Γ AODV p" by (metis aodv_ex_label)
  with assms(1) have "∀ dip ∈ kD (rt (σ i)). ?P dip"
    by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaodv_trans aodv_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 (opaodv 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 Γ AODV 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 Γ AODV (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 ai: "∀ 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))))"
\[(\text{the (nhop (rt (\sigma i)) dip)) \neq \text{dip}) \rightarrow rt (\sigma i) \sqsubseteq_d ip rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip))})\]

**proof clarify**

**fix dip**

assume a2: "dip\in D(rt (\sigma i))"

and a3: "dip\in D (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip))))"

and a4: "(\text{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_d ip rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip))))" by simp

**proof (cases "(\text{the (nhop (rt (\sigma i)) dip)) = i")**

assume "(\text{the (nhop (rt (\sigma i)) dip)) = i})" with \(dip \in D(rt (\sigma i))\) have "dip \in D(rt (\sigma (\text{the (nhop (rt (\sigma i)) dip)))))" by simp

with a1 a2 a4 have "rt (\sigma i) \sqsubseteq_d ip rt (\sigma (\text{the (nhop (rt (\sigma i)) dip))))" by simp

hence False by simp

thus ?thesis ..

**next**

assume "(\text{the (nhop (rt (\sigma i)) dip)) \neq i}" with \(\forall j. j \neq i \rightarrow \sigma j = \sigma' j\)

have \(*: \sigma (\text{the (nhop (rt (\sigma i)) dip)) = \sigma' (\text{the (nhop (rt (\sigma i)) dip)))" by simp

with \(dip \in D(rt (\sigma (\text{the (nhop (rt (\sigma i)) dip)))))\)

have "dip \in D(rt (\sigma (\text{the (nhop (rt (\sigma i)) dip)))))" by simp

with a1 a2 a4 have "rt (\sigma i) \sqsubseteq_d ip rt (\sigma (\text{the (nhop (rt (\sigma i)) dip))))" by simp

with * show ?thesis by simp

qed

**qed**

} note basic = this

\[
\{ \text{fix } \sigma \sigma' \ a \ dip \ sip \ i
\]

assume a1: "\forall dip. dip \in D(rt (\sigma i)) \land dip \in D (rt (\sigma (\text{the (nhop (rt (\sigma i)) dip)))) \land the (\text{nhop (rt (\sigma i)) dip}) \neq dip \rightarrow rt (\sigma i) \sqsubseteq_d ip rt (\sigma (\text{the (nhop (rt (\sigma i)) dip))})"

and ow: "?S i \sigma \sigma' a"

have "\forall dip. dip \in D(update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) \land dip \in D (rt (\sigma' (\text{the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip)))) \land the (\text{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, {}) \sqsubseteq_d ip rt (\sigma' (\text{the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip))))"

**proof clarify**

**fix dip**

assume a2: "dip\in D (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {}))"

and a3: "dip\in D (rt (\sigma' (\text{the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip)))))"

and a4: "(\text{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, {}) \sqsubseteq_d ip rt (\sigma' (\text{the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip))))" by simp

**proof (cases "dip = sip")**

assume "dip = sip"

with \(\text{the (nhop (update (rt (\sigma i)) sip (0, unk, val, Suc 0, sip, {})) dip}) \neq dip} \)

have False by simp

thus ?thesis ..

**next**

assume [simp]: "dip \neq sip"

from a2 have "dip\in D(rt (\sigma i)) \lor dip = sip"

by \(\text{rule vD_update_val})\)

with \(dip \neq sip) have "dip\in D(rt (\sigma i))" by simp

moreover from a3 have "dip\in D (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip)))))" by simp

moreover from a4 have "the (\text{nhop (rt (\sigma i)) dip}) \neq dip}" by simp

ultimately have "rt (\sigma i) \sqsubseteq_d ip rt (\sigma' (\text{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
\[ \{ \text{fix } \sigma \ a \ \sigma' \ nhop,\ \text{assume pre: } \exists \sigma' \text{ dip } \in vD (\text{rt } (\sigma i)) \land \text{dip } \in vD (\text{rt } (\sigma (\ text{nhop dip}))) \land \text{nhop dip } \neq \text{dip} \rightarrow \text{rt } (\sigma i) \sqsubseteq_{\text{dip}} \text{rt } (\sigma (\text{nhop dip}))) \]

and \text{ow: } "?S i \sigma \sigma' \ a" 

have "\forall \text{dip. dip } \in vD (\text{invalidate } (\text{rt } (\sigma i)) \land \text{dip } \in vD (\text{rt } (\sigma' \text{ (nhop dip) })) \land \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 "\text{dip } \in vD (\text{invalidate } (\text{rt } (\sigma i)) \land \text{dip } \in vD (\text{rt } (\sigma' \text{ (nhop dip) }) \land \text{nhop dip } \neq \text{dip}"

from this(1) have "\text{dip } \in vD (\text{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 "\text{rt } (\sigma i) \sqsubseteq_{\text{dip}} \text{rt } (\sigma (\text{nhop dip}))"

using pre \text{dip } \in vD (\text{rt } (\sigma' \text{ (nhop dip) })) \land \text{nhop dip } \neq \text{dip} by metis

with \forall j. j \neq i \rightarrow \sigma j = \sigma' j show "\text{rt } (\sigma i) \sqsubseteq_{\text{dip}} \text{rt } (\sigma' \text{ (nhop dip)})"

by (metis rt_strictly_fresher_irefl)

qed

}
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 sip))" 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
  moreove 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 = oip →
    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
    proof
assume \( "sqn (rt (\sigma i)) oip < osn" \)
also from \( osn \leq nsqn (rt (\sigma sip)) oip \)
have \( "... \leq nsqn (rt (\sigma i)) oip" \) by simp
also have \( "... \leq sqn (rt (\sigma i)) oip" \)
by (rule nsqn_sqn)
finally have \( "sqn (rt (\sigma i)) oip < sqn (rt (\sigma i)) oip" \).
hence False by simp
thus \( "(\lambda a. if a = oip then Some (osn, kno, val, Suc hops, i, \pi_7 (the (rt (\sigma i) oip))) else rt (\sigma i) a) = rt (\sigma i)" .. \)

next
assume \( "sqn (rt (\sigma i)) oip = osn" \)
and \( "Suc hops < the (dhops (rt (\sigma sip))) oip = osn" \)
by simp
with \( nsqn (rt (\sigma sip)) oip = osn \to the (dhops (rt (\sigma sip)) oip) \leq hops \)
have \( "the (dhops (rt (\sigma i)) oip) \leq hops" \) by simp
with \( Suc hops < the (dhops (rt (\sigma i)) oip) \) have False by simp
thus \( "(\lambda a. if a = oip then Some (osn, kno, val, Suc hops, i, \pi_7 (the (rt (\sigma i) oip))) else rt (\sigma i) a) = rt (\sigma i)" .. \)

next
assume \( "the (flag (rt (\sigma i)) oip) = inv" \)
with \( the (flag (rt (\sigma sip)) oip) = val \) have False by simp
thus \( "(\lambda a. if a = oip then Some (osn, kno, val, Suc hops, i, \pi_7 (the (rt (\sigma i) oip))) else rt (\sigma i) a) = rt (\sigma i)" .. \)

next
from \( oip \in kD(rt (\sigma sip)) \)
show \( "(\lambda a. if a = oip then Some (the (rt (\sigma i) oip)) else rt (\sigma i) a) = rt (\sigma i)" " \)
by (auto dest!: kD_Some)
qed

with change have False ..
thus \?thesis ..

qed

thus \?thesis
proof

assume \( osnlt: "osn < nsqn (rt (\sigma sip)) oip" \)

from \( dip \in kD(rt (\sigma i)) \) and \( dip = oip \) have "dip \in kD (?rt1)" by simp

moreover from \( a3 \) have "dip \in 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 (\sigma 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

note rreq_rrep_update = this

have "opaodv i |= (otherwith (op=) {i} (orecmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
               other quality_increases {i} →)
               onl Γ₉ΟΔV
(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 aodv_wf rreq_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf rrep_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf rerr_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf oosn_rreq [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf odsn_rrep [THEN weaken]]
onl_oinvariant_sterms [OF aodv_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 (opaodv i) (?S i) (other quality_increases {i})"
and "other quality_increases {i} σ'" and ll: "l ∈ labels Γ₉ΟΔV 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 (opaodv 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_oinspect_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_oinspect_weakenD [OF ozero_seq_unk_hops_one [OF oaodv_trans aodv_trans]])
otherwith_actionD
simp: sqnlaws

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 ⟨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 \( dip \in kD(rt (\sigma \ nhip)) \)
by (rule strictly_fresher_quality_increases_right)

next
assume "\( dip \notin vD(rt (\sigma \ nhip)) \"
with \( dip \in kD(rt (\sigma \ nhip)) \) have "\( dip \in iD(rt (\sigma \ nhip)) \)
therefore "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 \( dip \in vD(rt (\sigma \ i)) \) have "\( \ldots = sqn (rt (\sigma \ nhip)) dip - 1 \)
also have "\( \ldots < sqn (rt (\sigma' \ nhip)) dip \)
proof -
from \( \forall \sigma. quality_increases (\sigma \ i) (\sigma' \ j) \)
have "quality_increases (\sigma \ nhip) (\sigma' \ nhip)"
therefore "\( \forall \sigma. sqn (rt (\sigma \ i)) dip \leq sqn (rt (\sigma' \ i)) dip \)
by simp
finally have "\( nsqn (rt (\sigma \ i)) dip < nsqn (rt (\sigma' \ nhip)) dip \)".
moreover from \( dip \in vD(rt (\sigma' \ (the (nhop (rt (\sigma' \ i)) dip)))) \) and \( \sigma' \ i = \sigma \ i \)
ultimately show "\( rt (\sigma \ i) \sqsubseteq dip rt (\sigma' \ nhip) \)
using \( dip \in kD(rt (\sigma \ i)) \) by - (rule rt_strictly_fresher_ltI)
with \( \sigma' \ i = \sigma \ i \) show "\( rt (\sigma' \ i) \sqsubseteq dip rt (\sigma' \ (the (nhop (rt (\sigma' \ i)) dip))) \)
by simp
thus ?thesis unfolding Let_def .
qed

lemma seq_nhop_quality_increases:
shows "opaodv i \|= (otherwith (op=) \{i\})
\{orevmsg (\lambda m. msg_fresh (\sigma m) \land msg_zhops m),
other_quality_increases (\{\}) \rightarrow\}
\{global (\lambda \sigma. \forall dip. let nhip = the (nhop (rt (\sigma \ i)) dip)
in dip \in vD (rt (\sigma \ i)) \land vD (rt (\sigma \ nhip)) \land nhip \neq dip
\rightarrow (rt (\sigma \ i) \sqsubseteq dip rt (\sigma \ nhip))\}"
by (rule oinvariant_weakenE [OF seq_nhop_quality_increases']) (auto dest!: onlD)
end

0.11 Routing graphs and loop freedom

theory Loop_Freedom
imports Aodv_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 \Rightarrow state) \Rightarrow ip \Rightarrow ip rel"
where
"rt_graph \sigma = (\lambda dip.
\{(ip, ip') | ip ip' dsn dsk hops pre.
\ ip \neq dip \land 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 "$ip \neq dip \land (\exists r. rt (\sigma ip) = r$
  
  $\land (\exists dsn dsk hops pre. r dip = Some (dsn, dsk, val, hops, ip', pre))$"

Using asms unfolding rt_graph_def by auto

**Lemma rt_graph_vD [dest]:**

"$\forall ip ip' \sigma dip. (ip, ip') \in rt_graph \sigma dip \implies dip \in vD(rt (\sigma ip))$"

Unfolding rt_graph_def vD_def by auto

**Lemma rt_graph_vD_trans [dest]:**

"$\forall ip ip' \sigma dip. (ip, ip') \in rt_graph \sigma dip \land dip \in vD(rt (\sigma ip)) \implies dip \in vD(rt (\sigma ip))$"

By (erule converse_tranclE) auto

**Lemma rt_graph_not_dip [dest]:**

"$\forall ip ip' \sigma dip. (ip, ip') \in rt_graph \sigma dip \implies ip \neq dip$"

Unfolding rt_graph_def by auto

**Lemma rt_graph_not_dip_trans [dest]:**

"$\forall ip ip' \sigma dip. (ip, ip') \in rt_graph \sigma dip \land dip \neq dip \implies dip \neq dip$"

By (erule converse_tranclE) auto

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

**Lemma rt_graph_nhip_is_nhop [dest]:**

"$\forall ip ip' \sigma dip. (ip, ip') \in rt_graph \sigma dip \implies ip' = \text{the (nhop (rt (\sigma ip)) dip)}$"

Unfolding rt_graph_def by auto

**Theorem inv_to_loop_freedom:**

- assumes "$\forall i dip. let nhip = \text{the (nhop (rt (\sigma i)) dip)}$
  
  in $dip \in vD(rt (\sigma i)) \cap vD(rt (\sigma nhip)) \land nhip \neq dip$
  
  $\implies rt (\sigma i) \sqsubseteq dip rt (\sigma nhip)$"

- shows "$\forall dip. irrefl ((rt_graph \sigma dip)^+)$"

Using asms proof (intro allI)

- fix $\sigma :: \text{"ip \Rightarrow state" and dip}$

- assume inv: "$\forall ip dip.$
  
  let nhip = \text{the (nhop (rt (\sigma ip)) dip)}$
  
  in $dip \in vD(rt (\sigma ip)) \cap vD(rt (\sigma nhip)) \land nhip \neq dip \implies rt (\sigma ip) \sqsubseteq dip rt (\sigma nhip)$"

{ 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) \sqsubseteq 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 \in vD(rt (\sigma nhip))" ..

  with nhip = the (nhop (rt (\sigma ip)) dip)

  and nhip \neq dip

  and inv

  show "rt (\sigma ip) \sqsubseteq dip rt (\sigma nhip)"

  by (clarsimp simp: Let_def)

  next

  fix nhip nhip'

  assume "$(ip, nhip) \in (rt_graph \sigma dip)^+$"
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)"
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 Aodv_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 Γ AODV i) (seqp_sos Γ QMSG)"
  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 quality_increases], simp_all)
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 "opaodv 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])
qed auto

lemma par_rreq_rrep_nsqn_fresh_any_step:
shows "opaodv i \{(i qmsg |\_ A (λσ_, orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _ True) \{i\} →
globala (λ(σ, σ, a'). ancast (msg_fresh σ) a)"
proof -
have "opaodv i \|\_ A (λσ_, (orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _ True) \{i\} →
globala (λ(σ, σ, a'). ancast (msg_fresh σ) a)"
proof (rule ostep_invariant_weakenE [OF rreq_rrep_nsqn_fresh_any_step_invariant])
fix t
assume "onll Γ_AODV (λ((σ, _, _), a, _). ancast (msg_fresh σ) a) t"
thus "globala (λ(σ, a, σ'). ancast (msg_fresh σ) a) t"
by (cases t) (clarsimp dest!: onllD, metis aodv_ex_label)
qed auto

lemma par_anycast_msg_zhops:
shows "opaodv i \{(i qmsg |\_ A (λσ_, orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _ True) \{i\} →
globala (λ(_, a, _). ancast msg_zhops a)"
proof -
from anycast_msg_zhops_initial_aodv oadvt_trans aodvt_trans
have "opaodv i \|\_ A (act TT, other (λ_ _ True) \{i\} →
segll i (onll Γ_AODV (λ(_, a, _). ancast msg_zhops a))"
by (rule open_seq_step_invariant)
have "opaodv i \|\_ A (λσ_, orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _ True) \{i\} →
globala (λ(_, a, _). ancast msg_zhops a)"
proof (rule ostep_invariant_weakenE)
fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
assume "segll i (onll Γ_AODV (λ(_, a, _). ancast msg_zhops a)) t"
thus "globala (λ(_, a, _). ancast msg_zhops a) t"
by (cases t) (clarsimp dest!: segllD onllD, metis aodv_ex_label)
qed simp_all
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 "((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 : opaodv i ⟨⟨i qmsg : R⟩⟩o |=
      (otherwith (op=) {i} (oarrivemsg (λσ_. rreq_rrep_sn) σ, other (λ_ _. True) {i}) ∨ dip)
    global (λσ_. ∀dip. let nhip = the (nhop (rt (σ i)) dip)
      in dip ∈ vD (rt (σ i)) ⊑ dip (rt (σ nhip)) ∧ nhip ≠ dip
      → (rt (σ i)) □ dip (rt (σ nhip)))⟩= A ⟨⟨i qmsg : R⟩⟩o |=
      (otherwith (op=) {i} (oarrivemsg (λσ_. rreq_rrep_sn) σ, other (λ_ _. True) {i}) ∨ dip)
    globala (λ(σ, a, σ'). quality_increases (σ i) (σ' i))"
  by (rule node_lift [OF par_nhop_quality_increases]) auto

lemma node_quality_increases:
  shows "⟨⟨i : opaodv i ⟨⟨i qmsg : R⟩⟩o |= A ⟨⟨i qmsg : R⟩⟩o |=
      (other (λ_ _. True) {i}) ∨ dip)
    globala (λ(σ, a, σ'). quality_increases (σ i) (σ' i))⟩= A ⟨⟨i qmsg : R⟩⟩o |=
      (other (λ_ _. True) {i}) ∨ dip)
    globala (λ(σ, a, 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 : opaodv i ⟨⟨i qmsg : R⟩⟩o |=
      (other (λ_ _. True) {i}) ∨ dip)
    globala (λ(σ, a, σ'). castmsg (msg_fresh σ) a)⟩= A ⟨⟨i qmsg : R⟩⟩o |=
      (other (λ_ _. True) {i}) ∨ dip)
    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 : opaodv i ⟨⟨i qmsg : R⟩⟩o |=
      (other (λ_ _. True) {i}) ∨ dip)
    globala (λ(σ, a, σ'). castmsg msg_zhops a)⟩= A ⟨⟨i qmsg : R⟩⟩o |=
      (other (λ_ _. True) {i}) ∨ dip)
    globala (λ(σ, a, σ'). castmsg msg_zhops a)"
  by (rule node_lift_anycast_statelessassm [OF par_anycast_msg_zhops])

lemma node_silent_change_only:
  shows "⟨i : opaodv i ⟨⟨i qmsg : R⟩⟩o |=
      (other (λ_ _. True) {i}) ∨ dip)
    globala (λ(σ, a, σ'). a ≠ τ → σ' i = σ i)⟩= A ⟨⟨i qmsg : R⟩⟩o |=
      (other (λ_ _. True) {i}) ∨ dip)
    globala (λ(σ, a, σ'). a ≠ τ → σ' i = σ i)"
proof (rule ostep_invariantI, simp (no_asm), rule impl)
  fix σ a σ' ζ'
  assume or: "(σ, ζ) ∈ or reachable ((i : opaodv i ⟨⟨i qmsg : R⟩⟩o) |=
      (other (λ_ _. True) {i}) ∨ dip)
    globala (λ(σ, a, σ'). a ≠ τ → σ' i = σ i))"
  and tr: "((σ, ζ), a, (σ', ζ')) ∈ trans ((i : opaodv i ⟨⟨i qmsg : R⟩⟩o) |=
      (other (λ_ _. True) {i}) ∨ dip)
    globala (λ(σ, a, σ'). a ≠ τ → σ' i = σ i))" 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 (opaodv 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
0.12.3 Lift to partial networks

lemma arrive_rreq_rrep_msgn_fresh_inc_sn [simpl]:
  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. opaodv i ⟨⟨i qmsg : R⟩⟩) 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])
  have "((i : opaodv i ⟨⟨i qmsg : R⟩⟩) =_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 a σ' σ''
  assume or: "((σ, s) ∈ orachable ((i : opaodv i ⟨⟨i qmsg : R⟩⟩),
  (λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ)
  (other (λ_ _. True) {i})"
  and tr: "((σ, s), a, (σ', s')) ∈ trans ((i : opaodv i ⟨⟨i qmsg : R⟩⟩)"
  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_msgn_fresh_any_step])
  moreover from or tr am have "castmsg (msg_zhops m) 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 : opaodv i ⟨⟨i qmsg : R⟩⟩) =_A (λσ _. oarrivemsg (λ_. rreq_rrep_sn) 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 : opaodv i ⟨⟨i qmsg : R⟩⟩) =_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 ostepInvariant_WeakenE [OF node_silent_change_only]) auto
next
fix i R
show "(i : opaodv i ⟨⟨i qmsg : R⟩⟩) =_A (λσ _. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ),
  other_quality_increases {i} →) globala (λ(σ, a, _).
  a ≠ τ ∧ (∀i d. a ≠ i:deliver(d) → σ i = σ' i)"
  by (rule ostepInvariant_WeakenE [OF node_quality_increases]) auto
  simp_all

0.12.4 Lift to closed networks

lemma onet_nhop_quality_increases:
  shows "onet (opnet (λi. opaodv 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 "_ \models (\_, (?U \rightarrow) ?inv")

proof (rule inclosed_closed)
  from opnet_nhop_quality_increases
  show "opnet (\lambda i. opaodv i \langle i qmsg \rangle 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)
      \text{arrivemsg} (\lambda u m. msg_fresh \sigma m \land msg_zhops m)
    \sigma \sigma' a"
proof (rule otherwithEI)
  fix \sigma :: "ip \Rightarrow state"
  and a :: "msg node_action"
  assume 
    "inclosed \sigma a"
  thus 
    \text{arrivemsg} (\lambda u m. msg_fresh \sigma m \land msg_zhops m)
    \sigma a"
proof (cases a)
  fix ii ni ms
  assume 
    "a = ii \neg ni:arrive(ms)"
  moreover with 
    \langle inclosed \sigma a \rangle 
  obtain d di
  where 
    "ms = newpkt(d, di)"
  by (cases ms) auto
  ultimately show \?thesis by simp
qed simp_all
qed

0.12.5 Transfer into the standard model

interpretation aodv_openproc: openproc paodv opaodv id
where "aodv_openproc.initmissing = initmissing"
proof -
  show "openproc paodv opaodv id"
proof unfold_locales
  fix i :: ip
  have 
    "\{ (\sigma, \zeta). (\sigma i, \zeta) \in \sigma_{AODV} \land (\forall j. j \neq i \rightarrow \sigma j \in \text{fst } \sigma_{AODV} j) \} \subseteq \sigma_{AODV}," 
  unfolding \sigma_{AODV}_def \sigma_{AODV'}_def
  proof (rule equalityD1)
    show 
      \" \{ f p. \{ (\sigma, \zeta). (\sigma i, \zeta) \in \{(f i, p)\} \land (\forall j. j \neq i
      \rightarrow \sigma j \in \text{fst } \{(f j, p)\})\} = \{(f, p)\}\" 
    by (rule set_eqI) auto
  qed
  thus 
    \" \{ (\sigma, \zeta) \mid \sigma \zeta s. s \in \text{init (paodv i)}
    \land (\sigma i, \zeta) = \text{id s}
    \land (\forall j. j \neq i \rightarrow \sigma j \in \text{fst o id } \text{init (paodv j)}) \} \subseteq \text{init (opaodv i)}" 
  by simp
next
  show "\forall j. \text{init (paodv j)} \neq \{\}"
  unfolding \sigma_{AODV}_def by simp
next
  fix i s a s' \sigma \sigma'
  assume "\sigma i = \text{fst \text{id s)}"
  and "\sigma' i = \text{fst \text{id s')}" 
  and "(s, a, s') \in \text{trans (paodv i)}" 
  then obtain q q' where 
    "s = (\sigma i, q)"
  and "s' = (\sigma' i, q')" 
  and "((\sigma i, q), a, (\sigma' i, q')) \in \text{trans (paodv i)}" 
  by (cases s, cases s') auto
  from this(3) have "((\sigma, q), a, (\sigma', q')) \in \text{trans (opaodv i)}"
  by simp (rule open_seqp_action [OF aodv_wf])
with \langle s = (\sigma i, q) \rangle and \langle s' = (\sigma' i, q') \rangle
  show "((\sigma, \text{snd \text{id s)}), a, (\sigma', \text{snd \text{id s')}) \in \text{trans (opaodv i)}" 
  by simp
qed
then interpret op: openproc paodv opaodv id .
have [simp]: "∀ i. (SOME x. x ∈ (fst o id) ‘ init (paodv i)) = aodv_init i"

unfolding σ_AODV_def by simp

hence "∀ i. opnproc.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 aodv_openproc_par_qmsg: openproc_parq paodv opaodv id qmsg
where "aodv_openproc_par_qmsg.netglobal = netglobal"
and "aodv_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: "∀ σ. opnproc.initmissing (λ i. paodv 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 (λ i. paodv i ⟨⟨ qmsg ⟩⟩ (λ (p, q). (fst (id p), snd (id p), q)) = initmissing" by (rule ext)

have "∀ P. opnproc.netglobal (λ i. paodv 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_aodv_init_netlift [symmetric, unfolded initmissing_def])

thus "openproc.netglobal (λ i. paodv 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 (λ i. paodv 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 (λ i. paodv 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 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 σ i
assume sr: "σ ∈ reachable (closed (pnet (λ i. paodv i ⟨⟨ qmsg ⟩⟩ n) TT" hence "∀ i ∈ net_tree_ips n. ?inv (fst (initmissing (netgmap fst σ))) i"
by - (drule invariantD [OF proto],
simp only: aodv_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 (λ i. paodv 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

86
hence "(fst (initmissing (netgmap fst σ))) i = aodv_init i"
by simp
thus ?thesis by simp
qed metis
qed

0.12.6 Loop freedom of AODV

theorem aodv_loop_freedom:
assumes "wf_net_tree n"
shows "closed (pnet (λi. paodv i (⟨qmsg⟩ n) |= netglobal (λσ. ∀dip. irrefl ((rt_graph σ dip)⁺)))"
using assms by (rule aodv_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 originator's 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 ⇒ sqn"
  where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ 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 ≠ Suc 0"
  unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 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 × k × f × nat × ip × ip set"

definition proj2 :: "r ⇒ sqn" ("\(\pi_2\)"
  where "\(\pi_2 ≡ λ(dsn, _, _, _, _, _.). dsn\)"

definition proj3 :: "r ⇒ k" ("\(\pi_3\)"
  where "\(\pi_3 ≡ λ(_, dsk, _, _, _, _.). dsk\)"

definition proj4 :: "r ⇒ f" ("\(\pi_4\)"
where \( \pi_4 \equiv \lambda(\_\_, \_, \text{flag}, \_\_, \_\_). \text{flag} \)

**definition proj5** :: \( r \Rightarrow \text{nat} \) ("\( \pi_5 \)"
where \( \pi_5 \equiv \lambda(\_\_, \_, \text{hops}, \_\_, \_\_). \text{hops} \)

**definition proj6** :: \( r \Rightarrow \text{ip} \) ("\( \pi_6 \)"
where \( \pi_6 \equiv \lambda(\_\_, \_, \_, \text{nhip}, \_\_). \text{nhip} \)

**definition proj7** :: \( r \Rightarrow \text{ip set} \) ("\( \pi_7 \)"
where \( \pi_7 \equiv \lambda(\_\_, \_, \_, \_, \_\_, \text{pre}). \text{pre} \)

**lemma projs [simp]:**
\[
\begin{align*}
\pi_2 (dsn, dsk, flag, hops, nhip, pre) &= dsn \\
\pi_3 (dsn, dsk, flag, hops, nhip, pre) &= dsk \\
\pi_4 (dsn, dsk, flag, hops, nhip, pre) &= flag \\
\pi_5 (dsn, dsk, flag, hops, nhip, pre) &= hops \\
\pi_6 (dsn, dsk, flag, hops, nhip, pre) &= nhip \\
\pi_7 (dsn, dsk, flag, hops, nhip, pre) &= pre \\
\end{align*}
\]
by (clarsimp simp: proj2_def proj3_def proj4_def proj5_def proj6_def proj7_def)+

**lemma proj3_pred [intro]:** \([ \ P \ kno; P \ unk \] \implies P (\pi_3 x)\)
by (rule k.induct)

**lemma proj4_pred [intro]:** \([ \ P \ val; P \ inv \] \implies P (\pi_4 x)\)
by (rule f.induct)

**lemma proj6_pair_snd [simp]:**
fixes dsn' r
shows \( \pi_6 (dsn', \text{snd } (r)) = \pi_6 (r) \)
by (cases r) simp

### 1.1.3 Routing Tables

Routing tables map IP addresses to route entries.

**type synonym** \( rt = \text{"ip} \rightarrow r\) 

**syntax**
\[
\text{"_Sigma_route"} :: rt \Rightarrow \text{ip} \rightarrow r \ (\"\sigma\text{route}'(\_, \_')\"
\]

**translations**
\[
\"\sigma\text{route}(rt, dip)\" = \"rt dip\"
\]

**definition sqn ::** \( rt \Rightarrow \text{ip} \Rightarrow \text{sqn} \)
where \( \text{sqn } rt \ dip \equiv \text{case } \sigma\text{route}(rt, dip) \text{ of Some } r \Rightarrow \pi_2 (r) \mid \text{None } \Rightarrow 0 \)

**definition sqnf ::** \( rt \Rightarrow \text{ip} \Rightarrow k \)
where \( \text{sqnf } rt \ dip \equiv \text{case } \sigma\text{route}(rt, dip) \text{ of Some } r \Rightarrow \pi_3 (r) \mid \text{None } \Rightarrow \text{unk} \)

**abbreviation flag ::** \( rt \Rightarrow \text{ip} \rightarrow f \)
where \( \text{flag } rt \ dip \equiv \text{map_option } \pi_4 (\sigma\text{route}(rt, dip)) \)

**abbreviation dhops ::** \( rt \Rightarrow \text{ip} \rightarrow \text{nat} \)
where \( \text{dhops } rt \ dip \equiv \text{map_option } \pi_5 (\sigma\text{route}(rt, dip)) \)

**abbreviation nhop ::** \( rt \Rightarrow \text{ip} \rightarrow \text{ip} \)
where \( \text{nhop } rt \ dip \equiv \text{map_option } \pi_6 (\sigma\text{route}(rt, dip)) \)

**abbreviation precs ::** \( rt \Rightarrow \text{ip} \rightarrow \text{ip set} \)
where \( \text{precs } rt \ dip \equiv \text{map_option } \pi_7 (\sigma\text{route}(rt, dip)) \)

**definition vD ::** \( rt \Rightarrow \text{ip set} \)
where \( \text{vD } rt \equiv \{dip. \text{flag } rt \ dip = \text{Some } val\} \)
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.
    route 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 "route 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.
    route 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.
    route 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 \(\{\text{ip} \in iD(\text{rt})\}\) have \(\text{ip} \in kD(\text{rt})\) by auto
with assms show \(?\text{thesis}\) unfolding \(iD\text{.def}\) by auto
qed

lemma \(kD\text{.but_not_vD_is_iD}\) [elim]:
fixes \(\text{ip} \ \text{rt}\)
assumes \(\text{"ip} \in kD(\text{rt})\)"
and \(\text{"ip} \in vD(\text{rt})\)"
shows \(\text{"ip} \in iD(\text{rt})\)"
proof -
from \(\{\text{ip} \in kD(\text{rt})\}\) obtain \(dsn\) \(dsk\) \(f\) \(hops\) \(nhop\) \(pre\)
where \(\text{rtip} : \text{rt ip = Some (dsn, dsk, f, hops, nhop, pre)}\) by
(metis \(kD\text{.Some}\))
from \(\{\text{ip} \notin vD(\text{rt})\}\) have \(f \neq \text{val}\) proof
(rule contrapos nn)
assume \(f = \text{val}\)
with \(\text{rtip}\) have \(\text{the (flag rt ip)} = \text{val}\) by simp
with \(\{\text{ip} \in kD(\text{rt})\}\) show \(\text{ip} \in vD(\text{rt})\) ..
qed
with \(\text{rtip}\) have \(\text{the (flag rt ip)} = \text{inv}\) by simp
with \(\{\text{ip} \in kD(\text{rt})\}\) show \(\text{ip} \in iD(\text{rt})\) ..
qed

lemma \(vD\text{.or_iD}\) [elim]:
fixes \(\text{ip} \ \text{rt}\)
assumes \(\text{"ip} \in kD(\text{rt})\)"
and \(\text{ip} \in vD(\text{rt})\) = \(\Rightarrow \ P \text{ rt ip}\)"
and \(\text{ip} \in iD(\text{rt})\) = \(\Rightarrow \ P \text{ rt ip}\)
shows \(\text{P rt ip}\) proof -
from \(\{\text{ip} \in kD(\text{rt})\}\) have \(\text{ip} \in vD(\text{rt}) \cup iD(\text{rt})\) by
(simp add: \(kD\text{.is_vD_and_iD}\))
thus \(?\text{thesis}\) by (auto elim: assms(2-3))
qed

lemma \(\text{proj}_5\_eq_dhops\): \(\\text{\"\(\forall dip rt. dip} \in kD(rt)\) \Rightarrow \pi_5(\text{the (rt dip)}) = \text{the (dhops rt dip)}\)"
unfolding \(sqn\text{.def}\) by (drule \(kD\text{.Some}\)) clarsimp

lemma \(\text{proj}_4\_eq_flag\): \(\\text{\"\(\forall dip rt. dip} \in kD(rt)\) \Rightarrow \pi_4(\text{the (rt dip)}) = \text{the (flag rt dip)}\)"
unfolding \(sqn\text{.def}\) by (drule \(kD\text{.Some}\)) clarsimp

lemma \(\text{proj}_2\_eq_sqn\): \(\\text{\"\(\forall dip rt. dip} \in kD(rt)\) \Rightarrow \pi_2(\text{the (rt dip)}) = sqn rt dip\)"
unfolding \(sqn\text{.def}\) by (drule \(kD\text{.Some}\)) clarsimp

lemma \(kD\text{.sqn}_f\_is\_proj}_3\) [simp]:
\(\\text{\"\(\forall ip rt. ip} \in kD(rt)\) \Rightarrow sqnf rt ip = \pi_3(\text{the (rt ip)})\)"
unfolding \(sqnf\text{.def}\) by auto

lemma \(vD\text{.flag}_val\) [simp]:
\(\\text{\"\(\forall dip rt. dip} \in vD (rt) \Rightarrow \text{the (flag rt dip)} = \text{val}\)"
unfolding \(vD\text{.def}\) by clarsimp

lemma \(kD\text{.update}\) [simp]: \(\\text{\"\(\forall rt nip v. kD (rt(nip \mapsto v)) = insert nip (kD rt)\)"
unfolding \(kD\text{.def}\) by auto

lemma \(kD\text{.empty}\) [simp]: \(\text{\"kD Map.empty = \{\}\)"
unfolding \(kD\text{.def}\) by simp

lemma \(ip\_equal\_or\_known\) [elim]:
fixes \(\text{rt} \ \text{ip} \ \text{ip}'\)
assumes \(\text{"ip} = \text{ip}' \lor ip \in kD(\text{rt})\)"
and \(\text{"ip} = \text{ip}' \Rightarrow P \text{ rt ip ip}'\)"
and \(\ldelim{[}{18} \text{ip} \neq \text{ip}'; ip \in kD(\text{rt})\] \Rightarrow P \text{ rt ip ip}'\)"
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 "\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) ∪ 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)) (σ_{route}(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 \in kD rt"
and "ip' \in kD rt"
shows "\pi_2(\text{the (the (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 (the (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]:
  "\forall rt dip npre. dip \in kD(\text{rt}) \implies \pi_5(\text{the (the (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 "\text{flag (the (addpreRT rt dip pre)) ip} = \text{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 (\text{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 (\text{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 (\text{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 "\text{nhop (the (addpreRT rt dip pre)) ip} = \text{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 "\text{sqn (the (addpreRT rt dip pre)) ip} = \text{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 \in kD rt"
  shows "\text{dhops (the (addpreRT rt dip pre)) ip} = \text{dhops rt ip}"
  unfolding addpreRT_def

93
lemma sqnf_addpreRT [simp]:
"\( \forall \text{ip dip. ip} \in kD(\text{rt} \, \text{@}) \rightarrow \text{sqnf (the (addpreRT (\text{rt} \, \text{@}) \, \text{ip} \, \text{npre})}) \, \text{dip} = \text{sqnf (\text{rt} \, \text{@}) \, \text{dip}} \)
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 \text{rt dip of None } \Rightarrow \text{en } \mid \text{Some r } \Rightarrow \text{es r}) = \text{es (the (rt dip))}"
using assms [THEN kD_Some] by auto

lemma not_in_kD_case [simp]:
fixes dip rt
assumes "dip \not\in kD(rt)"
shows "(case \text{rt dip of None } \Rightarrow \text{en } \mid \text{Some r } \Rightarrow \text{es r}) = \text{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 \, \text{ip} = \text{Some (dsn, dsk, flag, hops, nhip, pre)}"
shows "\text{sqn rt ip} = \text{dsn}"
unfolding sqn_def using assms by simp

lemma not_kD_sqn [simp]:
fixes dip rt
assumes "dip \not\in kD(rt)"
shows "\text{sqn rt dip} = 0"
using assms unfolding sqn_def by simp

definition update_arg_wf :: "r \Rightarrow \text{bool}"
where "update_arg_wf r \equiv \pi_4(r) = \text{val} \land
(\pi_2(r) = 0) = (\pi_3(r) = \text{unk}) \land
(\pi_3(r) = \text{unk} \rightarrow \pi_5(r) = 1)"

lemma update_arg_wf_gives_cases:
"\( \forall \text{r. update_arg_wf r } \Rightarrow (\pi_2(r) = 0) = (\pi_3(r) = \text{unk}) \)"
unfolding update_arg_wf_def by simp

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

lemma update_arg_wf_tuples' [elim]:
"\( \forall \text{n hops nhip pre. Suc 0 \leq n } \Rightarrow \text{update_arg_wf (n, kno, val, hops, nhip, pre)} \)"
unfolding update_arg_wf_def by auto

lemma wf_r_cases [intro]:
fixes P r
assumes "\text{update_arg_wf r}"
and c1: "\( \forall \text{nhip pre. P (0, unk, val, Suc 0, nhip, pre)} \)"
and c2: "\( \forall \text{dsn hops nhip pre. dsn > 0 } \Rightarrow P (dsn, kno, val, hops, nhip, pre) \)"
shows "P r"
proof -
obtain dsn dsk flag hops nhip pre
where *: "r = (dsn, dsk, flag, hops, nhip, pre)" by (cases r)
with \text{update_arg_wf r} have wf1: "\text{flag = val}"
and wf2: "(dsn = 0) = (dsk = unk)"
and wf3: "dsk = unk \rightarrow (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 σ_route(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 if π3(r) = unk
      then rt (ip ↦→ addpre s (π7(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: "[\[ \begin{array}{l} ip \notin kD(rt) \end{array} \] \Rightarrow P (rt (ip ↦ r))]"
  and c2: "[\[ \begin{array}{l} ip \in kD(rt); sqn rt ip < π₂(r) \end{array} \] \Rightarrow P (rt (ip ↦ addpre r (π₇(the σroute(rt, ip))))])"
  and c3: "[\[ \begin{array}{l} ip \in kD(rt); sqn rt ip = π₂(r); the (dhops rt ip) > π₅(r) \end{array} \] \Rightarrow P (rt (ip ↦ addpre r (π₇(the σroute(rt, ip))))])"
  and c4: "[\[ \begin{array}{l} ip \in kD(rt); sqn rt ip = π₂(r); the (flag rt ip) = inv \end{array} \] \Rightarrow P (rt (ip ↦ addpre r (π₇(the σroute(rt, ip))))])"
  and c5: "[\[ \begin{array}{l} ip \in kD(rt); π₃(r) = unk \end{array} \] \Rightarrow P (rt (ip ↦ (π₃(the σroute(rt, ip)), π₃(r), π₄(r), π₅(r), π₆(r), π₇(addpre r (π₇(the σroute(rt, ip)))))), π₃(r))))"}
  and c6: "[\[ \begin{array}{l} ip \in kD(rt); sqn rt ip ≥ π₂(r); π₃(r) = kno; \end{array} \] \Rightarrow the (dhops rt ip) ≤ π₅(r) ∧ the (flag rt ip) = val]
  \Rightarrow 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 \( \text{thesis} \) using
\[
\begin{align*}
\pi_2(r) &= 0 = (\pi_3(r) = \text{unk}) \\
\text{c2: } &\langle \text{ip} \in kD(rt) \rangle \\
\text{c3: } &\langle \text{ip} \in kD(rt) \rangle \\
\text{c4: } &\langle \text{ip} \in kD(rt) \rangle \\
\text{c5: } &\langle \text{ip} \in kD(rt) \rangle \\
\text{c6: } &\langle \text{ip} \in kD(rt) \rangle
\end{align*}
\]
unfolding update_def sqn_def by auto

qed

lemma update_cases_kD:
assumes "\( \pi_2(r) = 0 \) = (\pi_3(r) = \text{unk})" and "\text{ip} \in kD(rt)" and c2: "sqn rt ip < \pi_2(r) \implies P (rt (ip \mapsto \text{addpre} r (\pi_7(\sigma_{\text{route}(rt, ip))))))" and c3: "[sqn rt ip = \pi_2(r); \text{dhops rt ip} > \pi_5(r)] \implies P (rt (ip \mapsto \text{addpre} r (\pi_7(\sigma_{\text{route}(rt, ip))))))" and c4: "[sqn rt ip = \pi_2(r); \text{flag rt ip} = \text{inv}] \implies P (rt (ip \mapsto \text{addpre} r (\pi_7(\sigma_{\text{route}(rt, ip))))))" and c5: "\pi_3(r) = \text{unk} \implies P (rt (ip \mapsto (\pi_2(\sigma_{\text{route}(rt, ip)}), \pi_3(r), \pi_4(r), \pi_5(r), \pi_6(r), \pi_7(\text{addpre} r (\pi_7(\sigma_{\text{route}(rt, ip))))))))" and c6: "[sqn rt ip \geq \pi_2(r); \pi_3(r) = \text{kno}; sqn rt ip = \pi_2(r) \implies \text{dhops rt ip} \leq \pi_5(r) \wedge \text{flag rt ip} = \text{val}] \implies P (rt (ip \mapsto \text{addpre} (\sigma_{\text{route}(rt, ip)})(\pi_7(r))))"

shows "\( P (\text{update rt ip r}) \)"
using assms(1) proof (rule update_cases)
assume "sqn rt ip < \pi_2(r)"
thus "P (rt(ip \mapsto \text{addpre} r (\pi_7(\text{the rt ip})))))" by (rule c2)
next
assume "sqn rt ip = \pi_2(r)"
and "the (dhops rt ip) > \pi_5(r)"
thus "P (rt(ip \mapsto \text{addpre} r (\pi_7(\text{the rt ip})))))" by (rule c3)
next
assume "sqn rt ip = \pi_2(r)"
and "the (flag rt ip) = \text{inv}"
thus "P (rt(ip \mapsto \text{addpre} r (\pi_7(\text{the rt ip})))))" 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), \pi_7(\text{addpre} r (\pi_7(\text{the } \sigma_{\text{route}(rt, ip)})))))))"
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{dhops rt ip} \leq \pi_5(r) \wedge \text{flag rt ip} = \text{val}"
thus "P (rt (ip \mapsto \text{addpre} (\text{the } \sigma_{\text{route}(rt, ip)})(\pi_7(r))))"
by (rule c6)
qed (simp add: \langle \text{ip} \in kD(rt) \rangle)

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

lemma nhop_of_update [simp]:
fixes rt dip dsn dsk flag hops nhip
assumes "rt \neq \text{update rt dip (dsn, dsk, flag, hops, nhip, \{}\)"
shows "\text{the} (\text{nhop} (\text{update rt dip (dsn, dsk, flag, hops, nhip, \{}\})) dip) = \text{nhip}"
proof -
from assms
have update_neq: "\\exists v. \text{rt dip} = \text{Some} v \implies"
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

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

lemma update_kno_dsn_greater_zero:
  "\rt dip 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 flg hops sip.
  rt ≠ update rt dip (dsn, dsk, flg, hops, sip, {})
  ⇒ π_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]: "\rt ip dsn dsk hops nhip.
\( rt \neq \text{update } rt \text{ ip (dsn, kno, val, hops, nhip, \{\})} \)
\( \Rightarrow \text{the (nhop (update } rt \text{ ip (dsn, kno, val, hops, nhip, \{\})) ip) = nhip} \)
using assms unfolding update_def
by (clarsimp split: option.split_asm option.split split_if_asm) auto

lemma flag_update [simp]: "\( rt \neq \text{update } rt \text{ dip (dsn, kno, flg, hops, sip, \{\})} \)
\( \Rightarrow \text{the (flag (update } rt \text{ dip (dsn, kno, flg, hops, sip, \{\})) dip) = flg} \)
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 \in \text{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 = \text{update } rt \text{ dip (dsn, dsk, flag, hops, nhip, pre)}"
  shows "dip \in \text{KD (rt)}"
proof -
  have "dip \in \text{KD (update rt dip (dsn, dsk, flag, hops, nhip, pre))}" by simp
  with assms show ?thesis by simp
qed

lemma nhop_update [simp]: "\( rt \neq \text{update } rt \text{ dip (dsn, dsk, flg, hops, sip, \{\})} \)
\( \Rightarrow \text{the (nhop (update } rt \text{ dip (dsn, dsk, flg, hops, sip, \{\})) dip) = sip} \)
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 \neq 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 \neq 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]:
  "\( \text{dip } rt \text{ dip'} (dsn, dsk, val, hops, nhip, pre) \Rightarrow (dip' \in \text{vD (rt)} \lor dip'=dip') \)"
unfolding update_def vD_def by (clarsimp split: option.split_asm split_if_asm)

Invalidating route entries

definition invalidate :: "rt \Rightarrow \nu (ip \Rightarrow \text{sqn} \Rightarrow rt)"
where "\text{invalidate } rt \text{ dests } \equiv \lambda \text{ip. case (rt ip, dests ip) of}
  (None, _) \Rightarrow \text{None}
  | (Some s, None) \Rightarrow \text{Some s}
  | (Some (_, dsk, _, hops, nhip, pre), Some rsn) \Rightarrow 
    \text{Some (rsn, dsk, inv, hops, nhip, pre)}"

lemma proj3_invalidate [simp]:
  "\( \text{\langle dip. } \pi_3(\text{the (invalidate rt dests) dip}) = \pi_3(\text{the (rt dip)}) \)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj5_invalidate [simp]:
"\(\forall \text{dip}. \pi_5(\text{the } (\text{invalidate rt dests}) \text{ dip}) = \pi_5(\text{the } (\text{rt dip})\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj6_invalidate [simp]:
"\(\forall \text{dip}. \pi_6(\text{the } (\text{invalidate rt dests}) \text{ dip}) = \pi_6(\text{the } (\text{rt dip})\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj7_invalidate [simp]:
"\(\forall \text{dip}. \pi_7(\text{the } (\text{invalidate rt dests}) \text{ dip}) = \pi_7(\text{the } (\text{rt dip})\)"
unfolding invalidate_def by (clarsimp split: option.split)

1.1.5 Route Requests

lemma invalidate_kD_inv [simp]:
"\(\forall \text{rt dests}. kD (\text{invalidate rt dests}) = kD \text{ rt}\)"
unfolding invalidate_def kD_def
by (simp split: option.split)

lemma invalidate_sqn:
fixes rt dip dests
assumes "\(\forall \text{rsn}. \text{dests dip } = \text{Some rsn} \longrightarrow \text{sqn rt dip } \leq \text{rsn}\)"
shows "\(\text{sqn rt dip } \leq \text{sqn } (\text{invalidate rt dests}) \text{ dip}\)"
proof (cases "\(\text{dip } \notin \text{kD(rt)}\)"
  assume "\(\neg \text{dip } \notin \text{kD(rt)}\)"
  hence "\(\text{dip } \in \text{kD(rt)}\)" by simp
  then obtain dsn dsk flag hops nhip pre where "\(\text{rt dip } = \text{Some } (dsn, dsk, flag, hops, nhip, pre)\)"
  by (metis kD_Some)
  with assms show "\(\text{sqn rt dip } \leq \text{sqn } (\text{invalidate rt dests}) \text{ dip}\)"
  by (cases "\(\text{dests dip}\)"
    (auto simp add: invalidate_def sqn_def)
qed simp

lemma sqn_invalidate_in_dests [simp]:
fixes dests ipa rsn rt
assumes "\(\text{dests ipa } = \text{Some rsn}\)"
  and "\(\text{ipa } \in \text{kD(rt)}\)"
  shows "\(\text{sqn } (\text{invalidate rt dests}) \text{ ipa } = \text{rsn}\)"
unfolding invalidate_def sqn_def
using assms(1) assms(2) [THEN kD_Some]
by clarsimp

lemma dhops_invalidate [simp]:
"\(\forall \text{dip}. \text{the } (\text{dhops } (\text{invalidate rt dests}) \text{ dip}) = \text{the } (\text{dhops rt dip})\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma sqnf_invalidate [simp]:
"\(\forall \text{dip}. \text{sqnf } (\text{invalidate } \xi (\text{dests } \xi)) \text{ dip} = \text{sqnf } (\text{rt } \xi) \text{ dip}\)"
unfolding sqnf_def invalidate_def by (clarsimp split: option.split)

lemma nhop_invalidate [simp]:
"\(\forall \text{dip}. \text{the } (\text{nhop } (\text{invalidate } \xi (\text{dests } \xi)) \text{ dip}) = \text{the } (\text{nhop } (\text{rt } \xi) \text{ dip})\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
fixes rt dests dip
assumes "\(\text{dip } \notin \text{dom(dests)}\)"
  shows "\(\text{invalidate rt dests dip } = \text{rt dip}\)"
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:
"\forall 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 ⇒ π_2(the (rt dip)) | Some rsn ⇒ rsn)
∧ dsk = π_3(the (rt dip))
∧ flag = (if dests dip = None then π_4(the (rt dip)) else inv)
∧ hops = π_5(the (rt dip))
∧ nhip = π_6(the (rt dip))
∧ pre = π_7(the (rt dip))"
using assms unfolding invalidate_def
by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: "\forall dip rt dests. dip∈kD (rt)
⟹ π_3(the (invalidate rt dests dip)) = π_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)

1.1.6 Queued Packets
Functions for sending data packets.

type_synonym store = "ip ⇒ (p × data list)"

definition sigma_queue :: "store ⇒ ip ⇒ data list" ("σ_queue\(_, _\)"
where "σ_queue\(store, dip\) ≡ 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 := None)
  | 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" ("σ_p-flag′(_, _)"")
where "σ_p-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 ≡ λ dip. 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. \( σ_p-flag(store, dip) \) is partial.
4. The routing table \( (rt) \) is modelled as a map \( (ip ⇒ r \ option) \) 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

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 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),
    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 ≠ 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),
    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

1.3.2 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 \( \text{is\_rreq} \ ξ \equiv \text{case msg} \ ξ \ of \)
\[
\text{Rreq hops' dip' dsn' dsk' oip' osn' sip'} \Rightarrow \\
\{ ξ | \text{hops := hops'}, dip := dip', dsn := dsn', \\
\text{dsk := dsk'}, oip := oip', osn := osn', sip := sip' } \}
\]
\| _ \Rightarrow \{ \}

lemma is\_rreq\_asm [dest!]:
assumes "ξ' ∈ is\_rreq ξ"
shows "(∃ hops' rreqid' dip' dsn' dsk' oip' osn' sip'.
msg ξ = Rreq hops' dip' dsn' dsk' oip' osn' sip' ∧
ξ' = ξ | hops := hops', dip := dip', dsn := dsn', \\
\text{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 \equiv \text{case msg} \ ξ \ of \\
\text{Rrep hops' dip' dsn' oip' sip'} \Rightarrow \\
\{ ξ | \text{hops := hops'}, dip := dip', dsn := dsn', \\
\text{oip := oip', sip := sip' } \}
\| _ \Rightarrow \{ \}

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', \\
\text{oip := oip', sip := sip' )"

using assms unfolding is\_rrep\_def
by (cases "msg ξ") simp_all

definition is\_err
where "is\_err ξ \equiv \text{case msg} \ ξ \ of \\
\text{Rerr dests' sip'} \Rightarrow \\
\{ ξ | \text{dests := dests'}, sip := sip' } \}
\| _ \Rightarrow \{ \}

lemma is\_err\_asm [dest!]:
assumes "ξ' ∈ is\_err ξ"
shows "(∃ dests' sip'.
msg ξ = Rerr dests' sip' ∧
ξ' = ξ | \text{dests := dests'}, sip := sip' )"

using assms unfolding is\_err\_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 \Longrightarrow \text{ip} ξ' = \text{ip} ξ"
"ξ' ∈ is\_rrep ξ \Longrightarrow \text{ip} ξ' = \text{ip} ξ"
"ξ' ∈ is\_rreq ξ \Longrightarrow \text{ip} ξ' = \text{ip} ξ"
"ξ' ∈ is\_pkt ξ \Longrightarrow \text{ip} ξ' = \text{ip} ξ"
"ξ' ∈ is\_newpkt ξ \Longrightarrow \text{ip} ξ' = \text{ip} ξ"

unfolding is\_msg\_defs
by (cases "msg ξ", clarsimp)+

lemma is\_msg\_inv\_sn [simp]:
"ξ' ∈ is\_rerr ξ \Longrightarrow \text{sn} ξ' = \text{sn} ξ"
"ξ' ∈ is\_rrep ξ \Longrightarrow \text{sn} ξ' = \text{sn} ξ"
"ξ' ∈ is\_rreq ξ \Longrightarrow \text{sn} ξ' = \text{sn} ξ"
"ξ' ∈ is\_pkt ξ \Longrightarrow \text{sn} ξ' = \text{sn} ξ"
"ξ' ∈ is\_newpkt ξ \Longrightarrow \text{sn} ξ' = \text{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+)+

1.3.3 The protocol process

datatype pseqp =
  PAodv | PNewPkt | PPkt | PRreq | PRrep | PRerr

fun nat_of_seqp :: "pseqp ⇒ 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 ≡
\[ \xi. \text{let } (\text{data}, \text{dip}, \text{oip}) = \text{args } \xi \text{ in}
\quad (\text{clear_locals } \xi) \quad (\text{data := data, dip := dip, oip := oip}) \]
\text{call}(\text{PPkt})";

abbreviation NEWPKT
where
"NEWPKT args ≡
\[ \xi. \text{let } (\text{data}, \text{dip}) = \text{args } \xi \text{ in}
\quad (\text{clear_locals } \xi) \quad (\text{data := data, dip := dip}) \]
\text{call}(\text{PNewPkt})";

abbreviation RREQ
where
"RREQ args ≡
\[ \xi. \text{let } (\text{hops}, \text{dip}, \text{dsn}, \text{dsk}, \text{oip}, \text{osn}, \text{sip}) = \text{args } \xi \text{ in}
\quad (\text{clear_locals } \xi) \quad (\text{hops := hops, dip := dip, dsn := dsn, dsk := dsk, oip := oip, osn := osn, sip := sip}) \]
\text{call}(\text{PRreq})";

abbreviation RREP
where
"RREP args ≡
\[ \xi. \text{let } (\text{hops}, \text{dip}, \text{dsn}, \text{oip}, \text{sip}) = \text{args } \xi \text{ in}
\quad (\text{clear_locals } \xi) \quad (\text{hops := hops, dip := dip, dsn := dsn, oip := oip, sip := sip}) \]
\text{call}(\text{PRrep})";

abbreviation RERR
where
"RERR args ≡
\[ \xi. \text{let } (\text{dests}, \text{sip}) = \text{args } \xi \text{ in}
\quad (\text{clear_locals } \xi) \quad (\text{dests := dests, sip := sip}) \]
\text{call}(\text{PRerr})";

fun Γₐ₀ᵈᵥ :: "\text{(state, msg, pseqp, pseqp label) seqp_env}" where
\[ Γₐ₀ᵈᵥ \text{PAodv} = \text{labelled PAodv} \]
\( \left\langle \text{receive}\lambda \text{msg' } \xi. \xi (\text{msg := msg'} ) \right\rangle \)
\( \quad \left\langle \text{(is_newpkt) NEWPKT(λξ. (data } \xi, \text{ip } \xi) } \right\rangle \)
\( \quad \left\langle \text{(is_pkt) PKT(λξ. (data } \xi, \text{dip } \xi, \text{oip } \xi) } \right\rangle \)
\( \quad \left\langle \text{(is_rreq) RREQ(λξ. (hops } \xi, \text{dip } \xi, \text{dsn } \xi, \text{dsk } \xi, \text{oip } \xi, \text{osn } \xi, \text{sip } \xi) } \right\rangle \)
\( \quad \left\langle \text{(is_rrep) RREP(λξ. (hops } \xi, \text{dip } \xi, \text{dsn } \xi, \text{oip } \xi, \text{sip } \xi) } \right\rangle \)
\( \quad \left\langle \text{(is_rerr) RERR(λξ. (dests } \xi, \text{sip } \xi) } \right\rangle \)
\( \quad \left\langle \lambda \xi. \{ \text{dip := dip }\} \mid \text{dip. dip } \in \text{qD(store } \xi) \cap \text{vD(rt } \xi) \} \right\rangle \)
\( \quad \left\langle \lambda \xi. \{ \text{data := hd(σ_queue(store } \xi, \text{ dip } \xi) ) } \right\rangle \)
\( \quad \text{unicast(λξ. the (nhop (rt } \xi) (dip )}, λξ. \text{pkt(data } \xi, \text{ dip } \xi, \text{ip } \xi)). \)
\( \quad \left\langle \lambda \xi. \{ \text{store := the (drop (dip ) (store } \xi) ) } \right\rangle \)
\( \quad \text{AODV()} \)
\( \quad \left\langle \lambda \text{ripr. if (rip } \in \text{vD (rt } \xi) \land \text{nhop (rt } \xi) \text{ rip } = \text{nhop (rt } \xi) \text{ (dip )) then Some (inc (sqn (rt } \xi) \text{ rip) else None}) } \right\rangle \)
\( \quad \left\langle \lambda \text{ripr. if ((dests } \xi) \text{ rip } \neq \text{None } \land \text{the (prec$s (rt } \xi) \text{ rip) } \neq \{} \right\rangle \)
\( \quad \text{then (dests } \xi) \text{ rip else None} \) \right\rangle \)
groupcast(λξ. pre ξ, λξ. rerr(dests ξ, ip ξ)). AODV()

Γ ΛPkt = labelled PNewPkt (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 ∪ {(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)). AODV()
"

Γ AODV PNewPkt = labelled PNewPkt (λξ. dip ξ = ip ξ)
   deliver(λξ. data ξ).AODV()

Γ AODV PPkt = labelled PPkt (λξ. dip ξ ≠ ip ξ)

Γ AODV PRreq = labelled PRreq (λξ. (oip ξ, osn ξ) ∈ rreqs ξ)
   AODV()

Γ AODV PRreq = labelled PRreq (λξ. (oip ξ, osn ξ) ∈ rreqs ξ)
   AODV()
[\zeta. \xi \mid \text{rt} := \text{the}\ (\text{addpreRT}\ (\text{rt} \xi)\ (\text{dip} \xi)\ \{\text{sip} \xi\})\ ]

[\zeta. \xi \mid \text{rt} := \text{the}\ (\text{addpreRT}\ (\text{rt} \xi)\ (\text{oip} \xi)\ \{\text{oip} \xi\})\ ]

unicast(\lambda\xi.\ \text{the}\ (\text{n-hop}\ (\text{rt} \xi)\ (\text{oip} \xi))\ ), \lambda\xi.\ \text{rrep}(\text{the}\ (\text{dhops}\ (\text{rt} \xi)\ (\text{dip} \xi))\ ),\ \text{dip} \xi, \text{sqn}\ (\text{rt} \xi)\ (\text{dip} \xi),\ \text{oip} \xi, \text{ip} \xi).

AODV()

\n
[\zeta. \xi \mid \text{dests} := \{\text{λrip. if}\ (\text{rip} \in \text{vd}\ (\text{rt} \xi) \land \text{n-hop}\ (\text{rt} \xi)\ \text{rip} = \text{n-hop}\ (\text{rt} \xi)\ (\text{oip} \xi))\}
\text{then}\ \text{Some}\ ((\text{inc}\ \text{sqn}\ (\text{rt} \xi)\ \text{rip}))\ \text{else}\ \text{None})\ ]

[\zeta. \xi \mid \text{rt} := \text{invalidate}\ (\text{rt} \xi)\ (\text{dests} \xi)\ ]

[\zeta. \xi \mid \text{store := setRRF}\ (\text{store} \xi)\ (\text{dests} \xi)\ ]

[\zeta. \xi \mid \text{pre} := \bigcup\ \{\ \text{dests}\ \text{rip} \in \text{dom}\ (\text{dests} \xi)\ \big\}\ ]

[\zeta. \xi \mid \text{dests} := \{\text{λrip. if}\ ((\text{dests} \xi)\ \text{rip} \neq \text{None} \land \text{the}\ (\text{precst}\ (\text{rt} \xi)\ \text{rip})\ \neq \{}\)
\text{then}\ (\text{dests} \xi)\ \text{rip} \ \text{else}\ \text{None})\ ]

\text{groupcast}(\lambda\xi.\ \text{pre} \xi, \lambda\xi.\ \text{rerr}(\text{dests} \xi, \text{ip} \xi)).\ \text{AODV}()

\n
\n
[\zeta. \xi \mid \text{dip} \xi \notin \text{vd}\ (\text{rt} \xi) \lor \text{sqn}\ (\text{rt} \xi)\ (\text{dip} \xi) < \text{dsn} \xi \lor \text{sqnf}\ (\text{rt} \xi)\ (\text{dip} \xi) = \text{unk})]

\text{broadcast}(\lambda\xi.\ \text{rreq}(\text{hops} \xi + 1,\ \text{dip} \xi,\ \text{max}\ \text{sqn}\ (\text{rt} \xi)\ (\text{dip} \xi))\ (\text{dsn} \xi),\ (\text{dsk} \xi,\ \text{oip} \xi,\ \text{osn} \xi,\ \text{ip} \xi))]

\text{AODV}()

\)

")

"\text{Γ}_{\text{AODV}}\ \text{PRrep} = \text{labelled}\ \text{PRrep}\ (\text{λrt} \xi \neq \text{update}\ (\text{rt} \xi)\ (\text{dip} \xi)\ (\text{dsn} \xi,\ \text{kno},\ \text{val},\ \text{hops} \xi + 1,\ \text{sip} \xi,\ \{\} ) )

[\zeta. \xi \mid \text{rt} := \text{update}\ (\text{rt} \xi)\ (\text{dip} \xi)\ (\text{dsn} \xi,\ \text{kno},\ \text{val},\ \text{hops} \xi + 1,\ \text{sip} \xi,\ \{\} ) ]

[\zeta. \xi \mid \text{oip} \xi = \text{ip} \xi )\ ]

\text{AODV}()

\n
\n
[\zeta. \xi \mid \text{oip} \xi \neq \text{ip} \xi )\ ]

[\zeta. \xi \mid \text{dests} := \{\text{λrip. if}\ (\text{rip} \in \text{vd}\ (\text{rt} \xi) \land \text{n-hop}\ (\text{rt} \xi)\ \text{rip} = \text{n-hop}\ (\text{rt} \xi)\ (\text{oip} \xi))\}
\text{then}\ \text{Some}\ ((\text{inc}\ \text{sqn}\ (\text{rt} \xi)\ \text{rip}))\ \text{else}\ \text{None})\ ]

[\zeta. \xi \mid \text{rt} := \text{invalidate}\ (\text{rt} \xi)\ (\text{dests} \xi)\ ]

[\zeta. \xi \mid \text{store := setRRF}\ (\text{store} \xi)\ (\text{dests} \xi)\ ]

[\zeta. \xi \mid \text{pre} := \bigcup\ \{\ \text{dests}\ \text{rip} \in \text{dom}\ (\text{dests} \xi)\ \big\}\ ]

[\zeta. \xi \mid \text{dests} := \{\text{λrip. if}\ ((\text{dests} \xi)\ \text{rip} \neq \text{None} \land \text{the}\ (\text{precst}\ (\text{rt} \xi)\ \text{rip})\ \neq \{}\)
\text{then}\ (\text{dests} \xi)\ \text{rip} \ \text{else}\ \text{None})\ ]

\text{groupcast}(\lambda\xi.\ \text{pre} \xi, \lambda\xi.\ \text{rerr}(\text{dests} \xi, \text{ip} \xi)).\ \text{AODV}()

\n
\n
[\zeta. \xi \mid \text{rt} \xi = \text{update}\ (\text{rt} \xi)\ (\text{dip} \xi)\ (\text{dsn} \xi,\ \text{kno},\ \text{val},\ \text{hops} \xi + 1,\ \text{sip} \xi,\ \{\} ) ]

\text{AODV}()

")

"\text{Γ}_{\text{AODV}}\ \text{PRerr} = \text{labelled}\ \text{PRerr}\ (\text{λdests} := \{\text{λrip. case}\ \text{dests} \xi\ \text{rip of}\ \text{None} \Rightarrow \text{None}\)
\text{\| Some}\ \text{rsn} \Rightarrow \text{if}\ \text{rip} \in \text{vd}\ (\text{rt} \xi) \land \text{the}\ (\text{n-hop}\ (\text{rt} \xi)\ \text{rip}) = \text{sip} \xi \land \text{sqn}\ (\text{rt} \xi)\ \text{rip} < \text{rsn}\ \text{then}\ \text{Some}\ \text{rsn}\ \text{else}\ \text{None})\ ]

[\zeta. \xi \mid \text{rt} := \text{invalidate}\ (\text{rt} \xi)\ (\text{dests} \xi)\ ]

[\zeta. \xi \mid \text{store := setRRF}\ (\text{store} \xi)\ (\text{dests} \xi)\ ]

[\zeta. \xi \mid \text{pre} := \bigcup\ \{\ \text{dests}\ \text{rip} \in \text{dom}\ (\text{dests} \xi)\ \big\}\ ]

[\zeta. \xi \mid \text{dests} := \{\text{λrip. if}\ ((\text{dests} \xi)\ \text{rip} \neq \text{None} \land \text{the}\ (\text{precst}\ (\text{rt} \xi)\ \text{rip})\ \neq \{}\)
\text{then}\ (\text{dests} \xi)\ \text{rip} \ \text{else}\ \text{None})\ ]

\text{groupcast}(\lambda\xi.\ \text{pre} \xi, \lambda\xi.\ \text{rerr}(\text{dests} \xi, \text{ip} \xi)).\ \text{AODV}()"
declare \( \Gamma_{AODV}.\text{simps} \) [simp del, code del]
lemmas \( \Gamma_{AODV}.\text{simps} \) [simp, code] = \( \Gamma_{AODV}.\text{simps} \) [simplified]

fun \( \Gamma_{AODV}.\text{skeleton} \)
where
  "\( \Gamma_{AODV}.\text{skeleton} \) PAodv = seqp_skeleton (\( \Gamma_{AODV} \) PAodv)"
  "\( \Gamma_{AODV}.\text{skeleton} \) PNewPkt = seqp_skeleton (\( \Gamma_{AODV} \) PNewPkt)"
  "\( \Gamma_{AODV}.\text{skeleton} \) PPkt = seqp_skeleton (\( \Gamma_{AODV} \) PPkt)"
  "\( \Gamma_{AODV}.\text{skeleton} \) PRreq = seqp_skeleton (\( \Gamma_{AODV} \) PRreq)"
  "\( \Gamma_{AODV}.\text{skeleton} \) PRrep = seqp_skeleton (\( \Gamma_{AODV} \) PRrep)"
  "\( \Gamma_{AODV}.\text{skeleton} \) PRerr = seqp_skeleton (\( \Gamma_{AODV} \) PRerr)"

lemma \( \Gamma_{AODV}.\text{skeleton}_\text{wf} \) [simp]:
  "wellformed \( \Gamma_{AODV}.\text{skeleton} \)"
proof (rule, intro allI)
  fix pn pn'
  show "call(pn') \notin \text{stermsl} \( \Gamma_{AODV}.\text{skeleton} \) pn"
  by (cases pn) simp_all
qed

declare \( \Gamma_{AODV}.\text{skeleton}.\text{simps} \) [simp del, code del]
lemmas \( \Gamma_{AODV}.\text{skeleton}.\text{simps} \) [simp, code] = \( \Gamma_{AODV}.\text{skeleton}.\text{simps} \) seqp_skeleton.simps

lemma aodv_proc_cases [dest]:
  fixes p pn
  shows "p \in \text{ctermsl} \( \Gamma_{AODV} \) pn" \( \equiv \)
  \( \bigvee \) (p \text{ \in } \text{ctermsl} \( \Gamma_{AODV} \) PAodv) \vee 
  p \text{ \in } \text{ctermsl} \( \Gamma_{AODV} \) PNewPkt \vee 
  p \text{ \in } \text{ctermsl} \( \Gamma_{AODV} \) PPkt \vee 
  p \text{ \in } \text{ctermsl} \( \Gamma_{AODV} \) PRreq \vee 
  p \text{ \in } \text{ctermsl} \( \Gamma_{AODV} \) PRrep \vee 
  p \text{ \in } \text{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 "\( \sigma_{AODV} \ i \equiv \{ \text{aodv_init } i, \Gamma_{AODV} PAodv \} \)"

abbreviation paodv
  :: "ip \Rightarrow (state \times (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
where
  "paodv i \equiv \{ \text{init } = \sigma_{AODV} i, \text{trans } = \text{seqp_sos } \Gamma_{AODV} \}"

lemma aodv_trans: "\text{trans } (\text{paodv } i) = \text{seqp_sos } \Gamma_{AODV}"
  by simp

lemma aodv_control_within [simp]: "\text{control_within } \Gamma_{AODV} (\text{init } (\text{paodv } i))"
  unfolding \( \sigma_{AODV}.\text{def} \) by (rule control_withinI) (auto simp del: \( \Gamma_{AODV}.\text{simps} \))

lemma aodv_wf [simp]:
  "wellformed \( \Gamma_{AODV} \)"
proof (rule, intro allI)
  fix pn pn'
  show "call(pn') \notin \text{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 \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 "\( \exists \ p \ l. \ P \ l \ p \Rightarrow Q \)"
shows "Q"
using assms by (metis aodv_ex_label)

lemma aodv_simple_labels [simp]: "simple_labels \( \Gamma_{AODV} \)"
proof
  fix \( \text{pn} \ p \)
  assume "\( p \in \text{subterms}(\Gamma_{AODV} \ \text{pn}) \)"
  thus "\( \exists! l. \ \text{labels} \ \Gamma_{AODV} \ p = \{l\} \)"
by (cases pn) (simp_all cong: seqp_congs | elim disjE)+
qed

lemma \( \sigma_{AODV} \_\text{labels} \) [simp]: "(\( \xi \), p) \in \sigma_{AODV} \ i \Rightarrow \ \text{labels} \ \Gamma_{AODV} \ p = \{\text{PAodv}-:0\}"
unfolding \( \sigma_{AODV} \_\text{def} \) by simp

lemma aodv_init_kD_empty [simp]:
"(\( \xi \), p) \in \sigma_{AODV} \ i \Rightarrow \ kD (\text{rt} \ \xi) = {}"
unfolding \( \sigma_{AODV} \_\text{def} \ kD \_\text{def} \) by simp

lemma aodv_init_sip_not_ip [simp]: "\neg (\text{sip} \ \text{aodv_init} \ i = i)" by simp

lemma aodv_init_sip_not_ip' [simp]:
assumes "(\( \xi \), p) \in \sigma_{AODV} \ i"
shows "\text{sip} \ \xi \neq \ \text{ip} \ \xi"
using assms unfolding \( \sigma_{AODV} \_\text{def} \) by simp

lemma aodv_init_sip_not_i [simp]:
assumes "(\( \xi \), p) \in \sigma_{AODV} \ i"
shows "\text{sip} \ \xi \neq i"
using assms unfolding \( \sigma_{AODV} \_\text{def} \) by simp

lemma clear_locals_sip_not_ip':
assumes "\text{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 seqp_congs [cong]

Configure the main invariant tactic for AODV.
declare \( \Gamma_{AODV} \_\text{simps} \) [cterms_env]
declare aodv_proc_cases [ctermsl_cases]
declare seq_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans,
cterms_intros]
declare seq_step_invariant_ctermsI [OF aodv_wf aodv_control_within aodv_simple_labels aodv_trans,
cterms_intros]

end

1.4 Invariant assumptions and properties

theory A_Aodv_Predicates
imports A_Aodv
define not_Pkt :: "msg \Rightarrow \text{bool}"
where "not_Pkt m \equiv \text{case} \ m \ of \ Pkt \ _ \ _ \ \Rightarrow \ False \ | \ _ \ \Rightarrow \ True"
definition msg_sender :: "msg \Rightarrow \text{ip}"

begin

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

where \( \text{msg\_sender } m \equiv \text{case } m \text{ of } \text{Rreq} \_ \_ \_ \_ \_ \_ \_ \_ \text{ipc} \Rightarrow \text{ipc} \\
\text{Rrep} \_ \_ \_ \_ \_ \text{ipc} \Rightarrow \text{ipc} \\
\text{Rerr} \_ \_ \_ \text{ipc} \Rightarrow \text{ipc} \\
\text{Pkt} \_ \_ \text{ipc} \Rightarrow \text{ipc} \)

lemma msg\_sender\_simps [simp]:

\[
\forall \text{hops dip dsn oip osn sip.} \quad \text{msg\_sender } (\text{Rreq hops dip dsn dip dip oip osn sip}) = \text{sip}
\]

\[
\forall \text{hops dip dsn oip sip.} \quad \text{msg\_sender } (\text{Rrep hops dip dsn oip sip}) = \text{sip}
\]

\[
\forall \text{dests sip.} \quad \text{msg\_sender } (\text{Rerr dests sip}) = \text{sip}
\]

\[
\forall \text{d dip sip.} \quad \text{msg\_sender } (\text{Pkt d dip sip}) = \text{sip}
\]

unfolding msg\_sender\_def by simp_all

definition msg\_zhops :: "msg \Rightarrow bool"

where "msg\_zhops m \equiv \text{case } m \text{ of } \text{Rreq} \text{ hopsc dipc } _ \_ \text{sipc} \Rightarrow \text{hopsc} = 0 \rightarrow \text{oipc} = \text{sipc} \\
\text{Rrep} \text{ hopsc dipc } _ \_ \_ \text{sipc} \Rightarrow \text{hopsc} = 0 \rightarrow \text{dipc} = \text{sipc} \\
_ \Rightarrow \text{True}"

lemma msg\_zhops\_simps [simp]:

\[
\forall \text{hops dip dsn oip osn sip.} \quad \text{msg\_zhops } (\text{Rreq hops dip dsn dip dip oip osn sip}) = (\text{hopsc} = 0 \rightarrow \text{oipc} = \text{sipc})
\]

\[
\forall \text{hops dip dsn oip sip.} \quad \text{msg\_zhops } (\text{Rrep hops dip dsn oip sip}) = (\text{hopsc} = 0 \rightarrow \text{dipc} = \text{sipc})
\]

\[
\forall \text{dests sip.} \quad \text{msg\_zhops } (\text{Rerr dests sip}) = \text{True}
\]

\[
\forall \text{d dip.} \quad \text{msg\_zhops } (\text{Newpkt d dip}) = \text{True}
\]

\[
\forall \text{d dip sip.} \quad \text{msg\_zhops } (\text{Pkt d dip sip}) = \text{True}
\]

unfolding msg\_zhops\_def by simp_all

definition rreq\_rrep\_sn :: "msg \Rightarrow bool"

where "rreq\_rrep\_sn m \equiv \text{case } m \text{ of } \text{Rreq} \_ \_ \_ \_ \_ \_ \_ \text{osnc} \Rightarrow \text{osnc} \geq 1 \\
\text{Rrep } \_ \_ \_ \_ \_ \_ \_ \text{dsnc} \Rightarrow \text{dsnc} \geq 1 \\
_ \Rightarrow \text{True}"

lemma rreq\_rrep\_sn\_simps [simp]:

\[
\forall \text{hops dip dsn oip osn sip.} \quad \text{rreq\_rrep\_sn } (\text{Rreq hops dip dsn dip dip oip osn sip}) = (\text{osnc} \geq 1)
\]

\[
\forall \text{hops dip dsn oip sip.} \quad \text{rreq\_rrep\_sn } (\text{Rrep hops dip dsn oip sip}) = (\text{dsnc} \geq 1)
\]

\[
\forall \text{dests sip.} \quad \text{rreq\_rrep\_sn } (\text{Rerr dests sip}) = \text{True}
\]

\[
\forall \text{d dip.} \quad \text{rreq\_rrep\_sn } (\text{Newpkt d dip}) = \text{True}
\]

\[
\forall \text{d dip sip.} \quad \text{rreq\_rrep\_sn } (\text{Pkt d dip sip}) = \text{True}
\]

unfolding rreq\_rrep\_sn\_def by simp_all

definition rreq\_rrep\_fresh :: "rt \Rightarrow msg \Rightarrow bool"

where "rreq\_rrep\_fresh crt m \equiv \text{case } m \text{ of } \text{Rreq} \_ \_ \_ \_ \_ \_ \_ \text{oipc osnc ipcc} \Rightarrow (\text{ipcc} \neq \text{oipc} \rightarrow \\
(\text{sqn} \text{ crt oipc} > \text{osnc} \\
\vee (\text{sqn} \text{ crt oipc} = \text{osnc} \\
\wedge \text{the (dhops crt oipc} \leq \text{hopsc} \\
\wedge \text{the (flag crt oipc} = \text{val}))))) \\
\text{Rrep} \text{ hopsc dipc dsnc ipcc} \Rightarrow (\text{ipcc} \neq \text{dipc} \\
\text{dipc} \in kD(crt) \\
\wedge \text{sqn} \text{ crt dipc} = \text{dsnc} \\
\wedge \text{the (dhops crt dipc} = \text{hopsc} \\
\wedge \text{the (flag crt dipc} = \text{val}) \\
_ \Rightarrow \text{True}"

lemma rreq\_rrep\_fresh\_simps [simp]:

\[
\forall \text{hops dip dsn oip osn sip.} \quad \text{rreq\_rrep\_fresh } (\text{Rreq hops dip dsn dip dip oip osn sip}) = (\text{sip} \neq \text{oip} \rightarrow \text{oip} \in kD(crt) \\
\vee (\text{sqn} \text{ crt oip} > \text{osn} \\
\wedge \text{the (dhops crt oip} \leq \text{hopsc} \\
\wedge \text{the (flag crt oip} = \text{val})))
\]

\[
\forall \text{hops dip dsn oip sip.} \quad \text{rreq\_rrep\_fresh } (\text{Rrep hops dip dsn oip sip}) = 
\]
\[
\begin{align*}
(sip \neq dip & \rightarrow dip \in kD(crt) \\
& \land sqn crt dip = dsn \\
& \land the (dhops crt dip) = hops \\
& \land the (flag crt dip) = val)"
\end{align*}
\]

"\land dests sip. \quad rreq_rrep_fresh crt (Rerr dests sip) = True"
"\land d dip. \quad rreq_rrep_fresh crt (Newpkt d dip) = True"
"\land d dip sip. \quad 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 = True"
| _ \Rightarrow True"

lemma rerr_invalid [simp]:
"\land hops dip dsn dsk oip osn sip. \quad rerr_invalid crt (Rreq hops dip dsn dsk oip osn sip) = True"
"\land hops dip dsn oip sip. \quad rerr_invalid crt (Rrep hops dip dsn oip sip) = True"
"\land dests sip. \quad rerr_invalid crt (Rerr dests sip) = (\forall rip \in dom(dests).
\quad rip \in iD(crt) \land the (dests rip) = sqn crt rip))
| _ \Rightarrow True"

lemma not_in_net_ips_fst_init_missing [simp]:
assumes "i /\in net_ips \sigma"
shows "fst (initmissing (netgmap fst \sigma)) i = aodv_init i | Some s \Rightarrow s, snd \sigma)"

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))
\quad = 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

1.5 Quality relations between routes

theory A_Fresher
imports A_Aodv_Data
begin

1.5.1 Net sequence numbers

On individual routes

definition
\( \text{nsqn} \cdot r \cdot s \Rightarrow q \)  

where  
\( \text{nsqn} \cdot r \equiv \text{if } \pi_4(r) = \text{val} \lor \pi_2(r) = 0 \text{ then } \pi_2(r) \text{ else } (\pi_3(r) - 1) \)  

**Lemma nsqn_r_def':**  
\( \text{nsqn} \cdot r = (\text{if } \pi_4(r) = \text{inv} \text{ then } \pi_2(r) - 1 \text{ else } \pi_2(r)) \)  

unfolding nsqn__def by simp

**Lemma nsqn_r_zero [simp]:**  
\( \forall dsn \ dsk \ flag \ hops \ nhip \ pre. \text{nsqn} \cdot r \cdot (dsn, dsk, flag, hops, nhip, pre) = 0 \)  

unfolding nsqn__def by clarsimp

**Lemma nsqn_r_val [simp]:**  
\( \forall dsn \ dsk \ hops \ nhip \ pre. \text{nsqn} \cdot r \cdot (dsn, dsk, val, hops, nhip, pre) = dsn \)  

unfolding nsqn__def by clarsimp

**Lemma nsqn_r_inv [simp]:**  
\( \forall dsn \ dsk \ hops \ nhip \ pre. \text{nsqn} \cdot r \cdot (dsn, dsk, inv, hops, nhip, pre) = dsn - 1 \)  

unfolding nsqn__def by clarsimp

**Lemma nsqn_r_lte_dsn [simp]:**  
\( \forall dsn \ dsk \ flag \ hops \ nhip \ pre. \text{nsqn} \cdot r \cdot (dsn, dsk, flag, hops, nhip, pre) \leq dsn \)  

unfolding nsqn__def by clarsimp

On routes in routing tables

**Definition**  
\( \text{nsqn} \cdot r \cdot \text{ip} \Rightarrow s \text{qn} \)  

where  
\( \text{nsqn} \equiv \lambda rt \ dip. \text{case } \sigma_{route}(rt, dip) \text{ of None } \Rightarrow 0 \ | \ \text{Some } r \Rightarrow \text{nsqn} \cdot r \cdot (r) \)

**Lemma nsqn_sqn_def:**  
\( \forall rt \ dip. \text{nsqn} \cdot r \cdot rt \ dip = (\text{if } flag \ rt \ dip = \text{Some val} \lor \text{sqn} \ rt \ dip = 0 \text{ then } \text{sqn} \ rt \ dip \text{ else } \text{sqn} \ rt \ dip - 1) \)  

unfolding nsqn_def sqn_def by (clarsimp split: option.split) auto

**Lemma not_in_kD_nsqn [simp]:**  
assumes "dip / \in \ 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 \cdot r \cdot (\sigma_{route}(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 (\text{nsqn} \cdot r \cdot (dsn, dsk, val, hops, nhip, pre))"  
and "P (\text{nsqn} \cdot r \cdot (dsn, dsk, inv, hops, nhip, pre))"  
shows "P (\text{nsqn} \cdot r \cdot (dsn, dsk, flag, hops, nhip, pre))"  
using assms by (cases flag) auto

**Lemma nsqnr__addpreRT_inv [simp]:**  
\( \forall rt \ dip \ npre \ dip'. \ dip \in kD(rt) \Rightarrow \text{nsqn} \cdot r \cdot (\text{the } (\text{addpreRT rt dip npre) dip'}) = \text{nsqn} \cdot r \cdot (\text{rt dip'}) \)  
unfolding addpreRT_def nsqn__def  
by (frule kD_Some) (clarsimp split: option.split)

**Lemma sqn_nsqn:**  
\( \forall rt \ dip. \text{sqn} \ rt \ dip - 1 \leq \text{nsqn} \ rt \ dip \)  
unfolding sqn_def nsqn_def by (clarsimp split: option.split)

**Lemma nsqn_sqn:**  "\text{nsqn} \ rt \ dip \leq \text{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]: 
\( \forall rt \ ip \ \dsn \ \dsk \ \hops \ nhip. \ rt \neq \ update \ rt \ ip \ (dsn, k\no, val, hops, nhip, \{\}) \implies \ nsqn \ (update \ rt \ ip \ (dsn, \dsk, val, \hops, nhip, \{\})) \ ip = \dsn \) 
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]: 
\( \forall rt \ dip \ npre \ dip'. \ dip \in kD(rt) \implies \ nsqn \ (the \ (addpreRT \ rt \ dip \ npre)) \ dip' = nsqn \ rt \ dip' \) 
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 ∈ 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)

1.5.2 Comparing routes

definition  
fresher :: "r ⇒ r ⇒ bool" ("(_/ ⊑ _)") [51, 51] 50
where
  "fresher r r' ≡ ((nsqn r < nsqn r') ∨ (nsqn r = nsqn r' ∧ π_5(r) ≥ π_5(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 "π_5(r) ≥ π_5(r')"
  shows "r ⊑ r'"
  unfolding fresher_def using assms by simp

lemma fresherI [intro]:
  assumes "(nsqn r r < nsqn r r') ∨ (nsqn r r = nsqn r r' ∧ π_5(r) ≥ π_5(r'))"
  shows "r ⊑ r'"
  unfolding fresher_def using assms.

lemma fresherE [elim]:
  assumes "r ⊑ r'"
  and "nsqn r r < nsqn r r' =⇒ P r r'"
  and "nsqn r r = nsqn r r' ∧ π_5(r) ≥ π_5(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'. ((σ_route(rt, dip)) ⊑ (σ_route(rt', dip)))"
abbreviation
\(rt\_fresher\_syn :: "rt \Rightarrow ip \Rightarrow rt \Rightarrow bool" \quad \text{("\_/ \_\_") \quad [51, 999, 51] 50}\)

where
\("rt1 \sqsubseteq rt2 \equiv rt\_fresher\ i\ rt1\ rt2\"

lemma \(rt\_fresher\_def':\)
\("(rt1 \sqsubseteq rt2) = (\text{nsqn}_{rt} (\text{the} (rt1 i)) < \text{nsqn}_{rt} (\text{the} (rt2 i)) \lor \\
\text{nsqn}_{rt} (\text{the} (rt1 i)) = \text{nsqn}_{rt} (\text{the} (rt2 i)) \land \pi_{5} (\text{the} (rt2 i)) \leq \pi_{5} (\text{the} (rt1 i)))"
\)

unfolding \(rt\_fresher\_def\ \text{fresher}\_\text{def}\ \text{by \ (rule \ refl)}\)

lemma \(\text{single}\_rt\_fresher\ \text{[intro]}:\)
assumes \("\text{the} (rt1 ip) \sqsubseteq \text{the} (rt2 ip)\"
shows \("rt1 \sqsubseteq \text{ip} \rt2\"
using \text{assms unfolding rt\_fresher\_def} .

lemma \(\text{rt\_fresher}\_\text{single} \text{[intro]}:\)
assumes \("rt1 \sqsubseteq \text{ip} \rt2\"
shows \("\text{the} (rt1 ip) \sqsubseteq \text{the} (rt2 ip)\"
using \text{assms unfolding rt\_fresher\_def} .

lemma \(\text{rt\_fresher}\_\text{def2}:\)
assumes \("\text{dip} \in kD(rt1)\"
and \("\text{dip} \in kD(rt2)\"
shows \("(rt1 \sqsubseteq dip \rt2) = (\text{nsqn rt1 dip} < \text{nsqn rt2 dip} \\
\lor \text{nsqn rt1 dip} = \text{nsqn rt2 dip} \\
\land \text{the (dhops rt1 dip)} \geq \text{the (dhops rt2 dip)})"
\)

using \text{assms unfolding rt\_fresher\_def2 \ [OF \ assms(1-2)] using assms(3) by simp}

lemma \(\text{rt\_fresher}\_\text{I1} \text{[intro]}:\)
assumes \("\text{dip} \in kD(rt1)\"
and \("\text{dip} \in kD(rt2)\"
and \("\text{nsqn rt1 dip} < \text{nsqn rt2 dip}\"
shows \("rt1 \sqsubseteq dip \rt2\"
unfolding \(rt\_fresher\_\text{def2} \ [OF \ assms(1-2)] using assms(3) by simp\)

lemma \(\text{rt\_fresher}\_\text{I2} \text{[intro]}:\)
assumes \("\text{dip} \in kD(rt1)\"
and \("\text{dip} \in kD(rt2)\"
and \("\text{nsqn rt1 dip} = \text{nsqn rt2 dip}\"
and \("\text{the (dhops rt1 dip)} \geq \text{the (dhops rt2 dip)}\"
shows \("rt1 \sqsubseteq dip \rt2\"
unfolding \(rt\_fresher\_\text{def2} \ [OF \ assms(1-2)] using assms(3-4) by simp\)

lemma \(\text{rt\_fresherE} \text{[elim]}:\)
assumes \("rt1 \sqsubseteq dip \rt2\"
and \("\text{dip} \in kD(rt1)\"
and \("\text{dip} \in kD(rt2)\"
and \("[\text{nsqn rt1 dip} < \text{nsqn rt2 dip} ] \implies P\ rt1\ rt2\ dip\"
and \("[\text{nsqn rt1 dip} = \text{nsqn rt2 dip} ; \\
\text{the (dhops rt1 dip)} \geq \text{the (dhops rt2 dip)} ] \implies P\ rt1\ rt2\ dip\"
shows \("P\ rt1\ rt2\ dip\"
using \text{assms(1) unfolding rt\_fresher\_def2 \ [OF \ assms(2-3)]}
using \text{assms(4-5) by auto}\)

lemma \(\text{rt\_fresher}\_\text{refl} \text{[simp]}: \ "rt} \sqsubseteq dip \ rt"
unfolding \(rt\_fresher\_\text{def} \text{by simp}\)

lemma \(\text{rt\_fresher\_trans} \text{[elim, trans]}:\)
assumes \("rt1 \sqsubseteq dip \rt2\"
and \("rt2 \sqsubseteq dip \rt3\"
shows \("rt1 \sqsubseteq dip \rt3\"
using \text{assms unfolding rt\_fresher\_def} \text{by auto}\)
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) ⊑ the (rt2 dip)"
  and "the (rt2 dip) ⊑ 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 "π₅(the (rt dip)) = π₅(the (rt' dip))"
  shows "rt ≈ dip rt'"
proof
  from assms(1-2,4) have dhops': "the (dhops rt' dip) ≤ the (dhops rt dip)"
  by (simp add: proj5_eq_dhops)
  with assms(1-3) show "rt ⊑ dip rt'"
  by (rule rt_fresherI2)
next
  from assms(1-2,4) have dhops: "the (dhops rt dip) ≤ the (dhops rt' dip)"
  by (simp add: proj5_eq_dhops)
  with assms(2,1) assms(3) [symmetric] show "rt' ⊑ dip rt"
  by (rule rt_fresherI2)
qed

lemma rt_fresh_asE [elim]:
  assumes "rt1 ≈ dip rt2"
  and "[ rt1 ⊑ dip rt2; rt2 ⊑ dip rt1 ] ⇒ 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 ≈ dip rt2"
  shows "rt1 ⊑ dip rt2"
using assms unfolding 
rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
  assumes "rt1 ≈ dip rt2"
  shows "rt2 ⊑ dip rt1"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
  assumes "rt1 ≈ dip rt2"
  shows "rt2 ≈ dip rt1"
using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
  assumes "¬ (rt1 ⊑ dip rt2)"
  shows "¬ (rt1 ≈ dip rt2)"
proof
  assume "rt1 ≈ dip rt2"
  hence "rt1 ⊑ dip rt2" ..
  with ⟨¬ (rt1 ⊑ dip rt2)⟩ show False ..
qed

lemma not_rt_fresh_asI2 [intro]:
  assumes "¬ (rt2 ⊑ dip rt1)"
  shows "¬ (rt1 ≈ dip rt2)"
proof
  assume "rt1 ≈ dip rt2"
  hence "rt2 ⊑ dip rt1" ..
  with ⟨¬ (rt2 ⊑ 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 "nsqn r (the (rt2 dip)) = nsqn r (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 "dhops rt dip ≥ 1"
 and "update_arg_wf r"
 shows "rt ⊑ dip update rt dip 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)⟩
 have [simp]: "sqn rt dip = dsn"
 and [simp]: "the (dhops rt dip) = hops"
 and [simp]: "the (flag rt dip) = f"
 by (simp add: sqn_def proj5_eq_dhops [symmetric] proj4_eq_flag [symmetric])+

from ⟨update_arg_wf r⟩ have "(dsn, dsk, f, hops, nhip, pre) ⊑ the ((update rt dip r) dip)"

proof (rule wf_r_cases)
 fix nhip pre
 from ⟨hops ≥ 1⟩
 have "\pre'. (dsn, dsk, f, hops, nhip, pre) ⊑ (dsn, unk, val, Suc 0, nhip, pre')"
 unfolding fresher_def sqn_def by (cases f) auto
 thus "(dsn, dsk, f, hops, nhip, pre) ⊑ 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 "(dsn, dsk, f, hops, nhip, pre) ⊑ 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 "dsn < dsn"
 thus "(dsn, dsk, f, hops, nhip, pre) ⊑ (dsn, kno, val, hops, nhip, pre ∪ pre)"
 unfolding fresher_def by auto

next
 assume "dsn = dsn"
 and "hops < hops"
 thus "(dsn, dsk, f, hops, nhip, pre) ⊑ (dsn, kno, val, hops, nhip, pre ∪ pre)"
unfolding fresher_def nsqn_r_def by simp

next
assume "dsn_n = dsn"
with (0 < dsn)
show "(dsn, dsk_n, inv, hops_n, nhip_n, pre_n)
    ⊆ (dsn, kno, val, hops, nhip, pre ∪ pre_n)"
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 instests: "∀ 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 instests 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 "nsqn_r (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 ⟨dip ∈ vD(rt)⟩ have "sqn rt dip = 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_r_fresh_asI)
from ⟨dip ∈ vD(rt)⟩ have "nsqn rt dip = sqn rt dip" by simp
also have "sqn rt dip = nsqn_r (the (invalidate rt dests dip))"
proof -
from ⟨dip ∈ kD(rt)⟩ have "nsqn_r (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 = nsqn_r (the (invalidate rt dests dip))" by simp
qed
also from \(dip \in \kD\) (invalidate rt dests),

have

\[
\text{"nsqn}_r \text{ (the (invalidate rt dests dip))} = \text{nsqn} \text{ (invalidate rt dests dip)}
\]

by (simp add: \(kD\_nsqn\))

finally show

\[
\text{"nsqn rt dip = nsqn (invalidate rt dests dip)\" .}
\]

qed simp

lemmas rt_fresher_inc_invalidate [simp] = rt_fresh_as_inc_invalidate [THEN rt_fresh_asD1]

lemma rt_fresh_as_addpreRT [simp]:

assumes \(\text{"ip} \in \kD\text{ (rt)\"}\)

shows \(\text{"rt} \approx dip \text{ (addpreRT rt ip npre)\"}\)

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 \Rightarrow rt \Rightarrow rt \Rightarrow bool"

where

\[
\text{"rt}\_\text{strictly}\_\text{fresher} \equiv \lambda dip rt1 rt2. (rt1 \sqsubseteq dip rt2) \land \neg (rt1 \approx dip rt2)"
\]

abbreviation

\[
\text{rt}\_\text{strictly}\_\text{fresher}\_\text{syn} :: \text{"rt} \Rightarrow \text{ip} \Rightarrow rt \Rightarrow bool\" \text{\("/ \sqsubseteq \_\_\text{\")\"} \text{[51, 999, 51] 50}\]

where

\[
\text{"rt1} \sqsubseteq dip \text{ rt2 \equiv rt}\_\text{strictly}\_\text{fresher} i \text{ rt1} \text{ rt2}\"
\]

lemma rt_strictly_fresher_def'':

\[
\text{"rt1} \sqsubseteq dip \text{ rt2} = ((\text{nsqn} r \text{ (the (rt1 i))} < \text{nsqn} r \text{ (the (rt2 i))})
\]

\[
\lor (\text{nsqn} r \text{ (the (rt1 i))} = \text{nsqn} r \text{ (the (rt2 i))} \land \pi_5(\text{the (rt1 i)}) > \pi_5(\text{the (rt2 i)})))\"

unfolding rt_strictly_fresher_def'' rt_fresher_def by auto

lemma rt_strictly_fresherI' [intro]:

assumes \(\text{"rt1} \sqsubseteq dip \text{ rt2}\"

and \(\neg (\text{rt2} \sqsubseteq dip \text{ rt1})\)

shows \(\text{"rt1} \sqsubseteq dip \text{ rt2}\"

using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherE' [elim]:

assumes \(\text{"rt1} \sqsubseteq dip \text{ rt2}\"

and \(\neg (\text{rt2} \sqsubseteq dip \text{ rt1})\)

shows \(\text{P rt1 rt2 i}\"

using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherI [intro]:

assumes \(\text{"rt1} \sqsubseteq dip \text{ rt2}\"

and \(\neg (\text{rt1} \approx dip \text{ rt2})\)

shows \(\text{rt1} \sqsubseteq dip \text{ 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 \(\text{"rt1} \sqsubseteq dip \text{ rt2}\"

and \(\neg (\text{rt1} \approx dip \text{ rt2})\)

shows \(\text{P rt1 rt2 i}\"

using assms(1) unfolding rt_strictly_fresher_def
by rule (erule(1) assms(2))

lemma rt_strictly_fresher_def':

\[
\text{rt1} \sqsubseteq dip \text{ rt2} =
\]

\[
(\text{nsqn} r \text{ (the (rt1 i))} < \text{nsqn} r \text{ (the (rt2 i))} \\
\lor (\text{nsqn} r \text{ (the (rt1 i))} = \text{nsqn} r \text{ (the (rt2 i))} \land \pi_5(\text{the (rt1 i)}) > \pi_5(\text{the (rt2 i)})))\"

unfolding rt_strictly_fresher_def'' rt_fresher_def_def by auto
lemma rt_strictly_fresher_fresherD [dest]:
  assumes "rt1 ⊑ dip rt2"
  shows "the (rt1 dip) ⊑ 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 ⊏ dip rt2"
  shows "¬ rt1 ≈ dip rt2"
  using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
  assumes "rt1 ⊏ dip rt2"
  and "rt2 ⊑ dip rt3"
  shows "rt1 ⊏ dip rt3"
  using assms proof -
  from ⟨rt1 ⊏ dip rt2⟩ obtain "the (rt1 dip) ⊑ the (rt2 dip)" by auto
  also from ⟨rt2 ⊑ dip rt3⟩ obtain "the (rt2 dip) ⊑ the (rt3 dip)" by auto
  finally have "the (rt1 dip) ⊑ the (rt3 dip)".
  moreover have "¬ (rt1 ≈ dip rt3)"
  proof -
  from ⟨rt1 ⊏ dip rt2⟩ obtain "¬ (the (rt2 dip) ⊑ the (rt1 dip))" by auto
  also from ⟨rt2 ⊑ dip rt3⟩ obtain "¬ (the (rt3 dip) ⊑ the (rt2 dip))" by auto
  finally have "¬ (the (rt3 dip) ⊑ the (rt1 dip))".
  thus thesis ..
  qed
  ultimately show "rt1 ⊏ dip rt3" ..
  qed

lemma rt_strictly_fresher_irefl [simp]: "¬ (rt ⊏ dip rt)"
  using assms unfolding rt_strictly_fresher_def by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elimsimp]
  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 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 rt1)" have "¬ (rt3 ⊑ dip rt1)"
  qed
proof (rule contrapos_neg)
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 rt') = (rt ⊑ dip rt')"[nsqn]
unfolding invalidate_def rt_strictly_fresher_def' by (auto simp add: kd_nsqn)

lemma invalidate_vD_def [simp]:
assumes "dip ∈ vD(invalidate rt1 dests)"
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
ultimately show \( \Rightarrow \) thesis

unfolding invalidate_def rt_strictly_fresher_def' by auto

qed

lemma rt_strictly_fresher_update_other [elim!]:
"(\exists ip rt r rt'. [ \( \neg \) dip \( \rightarrow \) r rt']. \( \Rightarrow \) update rt ip r dip rt' )"

unfolding rt_strictly_fresher_def' by clarsimp

lemma addpreRT_strictly_fresher [simp]:
assumes "\( dip \in \text{kd}(rt) \)"
shows "(the (addpreRT rt dip npre) \( \cap \) ip rt2) = (rt \( \cap \) ip rt2)"

using assms unfolding rt_strictly_fresher_def' by clarsimp

lemma lt_sqn_imp_update_strictly_fresher:
assumes "\( dip \in \text{vD}(rt2 nhip) \)"
and â*: "osn < sqn (rt2 nhip) dip"
and â**: "rt \( \neq \) update rt dip (osn, kno, val, hops, nhip, {})"
shows "update rt dip (osn, kno, val, hops, nhip, {}) \( \cap \) dip rt2 nhip"

unfolding rt_strictly_fresher_def'
proof (rule disjI1)
from â** have "nsqn (update rt dip (osn, kno, val, hops, nhip, {})) dip = osn"
  by (rule nsqn_update_changed_kno_val)
with \( \langle dip \in \text{vd}(rt2 nhip) \rangle \)
  have "nsqn, (the (update rt dip (osn, kno, val, hops, nhip, {})) dip)) = osn"
  by (simp add: \text{kd} nsqn)
also have "osn < sqn (rt2 nhip) dip" by (rule â*)
also have "sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))"
unfolding nsqn_def using \( \langle dip \in \text{vd}(rt2 nhip) \rangle \)
by - (metis \text{vd_flag_val proj2_eq_sqn proj4_eq_flag \text{vD_id_gives_kD}(1))
finally show "nsqn, (the (update rt dip (osn, kno, val, hops, nhip, {})) dip)) < nsqn, (the (rt2 nhip dip))".

qed

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

unfolding rt_strictly_fresher_def'
proof (rule disjI2, rule conjI)
from â** have "nsqn (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip = osn"
  by (rule nsqn_update_changed_kno_val)
with \( \langle dip \in \text{vd}(rt2 nhip) \rangle \)
  have "nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)) = osn"
  by (simp add: \text{kd} nsqn)
also have "osn = sqn (rt2 nhip) dip" by (rule \text{sqn [symmetric]})
also have "sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))"
unfolding nsqn_def using \( \langle dip \in \text{vd}(rt2 nhip) \rangle \)
by - (metis \text{vd_flag_val proj2_eq_sqn proj4_eq_flag \text{vD_id_gives_kD}(1))
finally show "nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip)) = nsqn, (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 â** 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)) \leq \pi_5 (the (update rt dip (osn, kno, val, Suc hops, nhip, {})) dip))" using \( \langle dip \in \text{vd}(rt2 nhip) \rangle \)
  by (simp add: \text{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 Γ_AODV (λ((ξ, _), _). sn ξ ≤ sn ξ')"
by inv_cterms

lemma sequence_number_one_or_bigger:
"paodv i |= onl Γ_AODV (λ(ξ, _). 1 ≤ sn ξ)"
by (rule onll_step_to_invariantI [OF sequence_number_increases]) (auto simp: σ_AODV_def)

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

lemma sequence_number_increases':
"paodv i |= A (λ((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 Γ_AODV (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 Γ_AODV (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 Γ_AODV (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 Γ_AODV (λ(ξ, l). ∀dip∈kD(rt ξ). the (nhop (rt ξ) dip)∈kD(rt ξ))"
proof -
{ fix ip and \( \xi, \xi' :: \text{state} \)
assume \( \forall \) dip \( \in \kD (rt \xi). \) the (nhop (rt \xi) dip) \( \in \kD (rt \xi) \)
and \( \xi' = \xi[rt := update (rt \xi) ip \{0, unk, val, Suc 0, ip, \}] \)
}
by clarsimp (metis nhop_update_unk_val update_another)
}

note one_hop = this
{ fix ip sip sn hops and \( \xi, \xi' :: \text{state} \)
assume \( \forall dip \in \kD (rt \xi). \) 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 \( \forall dip \in \kD (rt \xi). \) the (nhop (update (rt \xi) ip \{sn, kno, val, Suc hops, sip, \}) dip) \( \in \kD (rt \xi) \)
\( \lor \) 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 aodv_wf sip_in_kD] onl_invariant_sterms [OF aodv_wf addpreRT_partly_welldefined] solve: one_hop nhip_is_sip)
qed

Proposition 7.22: needed in Proposition 7.4
lemma addpreRT_welldefined:
"paodv i \parallel= \onl \Gamma_{AODV} (\lambda(\xi, 1). \( \{ PRreq-:16..PRreq-:18 \} \longrightarrow dip \xi \in \kD (rt \xi) \) \land
( l = PRreq-:17 \longrightarrow oip \xi \in \kD (rt \xi) ) \land
( l = PRreq-:6 \longrightarrow dip \xi \in \kD (rt \xi) ) \land
( l = PRreq-:6 \longrightarrow (the (nhop (rt \xi) (dip \xi))) \in \kD (rt \xi)) )"
(is "_ \parallel= \onl \Gamma_{AODV} ?P")

unfolding invariant_def
proof
fix s
assume "s \in reachable (paodv i) TT"
then obtain \( \xi, p \) where "s = (\xi, p)"
and "(\xi, p) \in reachable (paodv i) TT"
by (metis PairE)

have "onl \Gamma_{AODV} ?P (\xi, p)"
proof (rule onlI)
fix onlI
fix l
assume "l \in labels \Gamma_{AODV} p"
with \( (\xi, p) \in reachable (paodv i) TT \) have I1: "l \in \{ PRreq-:16..PRreq-:18 \} \longrightarrow dip \xi \in \kD (rt \xi)"
and I2: "l = PRreq-:17 \longrightarrow oip \xi \in \kD (rt \xi)"
and I3: "l \in \{ PRreq-:2..PRreq-:6 \} \longrightarrow dip \xi \in \kD (rt \xi)"
by (auto dest!: invariantD [OF addpreRT_partly_welldefined])

moreover from \( (\xi, p) \in reachable (paodv i) TT \) \( l \in labels \Gamma_{AODV} p \) and I3
have "l = PRreq-:6 \longrightarrow (the (nhop (rt \xi) (dip \xi))) \in \kD (rt \xi)"
by (auto dest!: invariantD [OF includes_nhip])

ultimately show "?P (\xi, 1)"
by simp
qed

with \( s = (\xi, p) \) show "onl \Gamma_{AODV} ?P s"
by simp
qed

Proposition 7.4
lemma known_destinations_increase:
"paodv i \models_A onl \Gamma_{AODV} (\lambda(\xi, _), _, (\xi', _)). \kD (rt \xi) \subseteq \kD (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:**

\[ \text{paodv } i \models A \Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \text{rreqs } \xi \subseteq \text{rreqs } \xi') \]

by \text{inv_cterms simp add: subset_insertI}

**Lemma dests_bigger_than_sqn:**

\[ \text{paodv } i \models \text{onl } \Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \text{rreqs } \xi \subseteq \text{rreqs } \xi') \]

proof -
have sqninv:
  
  "\forall \text{dests } rt \text{ rsn } ip. 
  \forall ip \in \text{dom}(\text{dests}). \text{ip} \in \text{kD}(rt) \land \text{sqn } \text{rt } \text{ip} \leq \text{the (dests } \text{ip}) \]

  \[ \Rightarrow \text{sqn (invalidate } \text{rt } \text{dests) } \text{ip} \leq \text{rsn} \]
  
  by \text{rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption} auto

have indests:
  "\forall \text{ip} \in \text{dom}(\text{dests}). \text{ip} \in \text{kD}(\text{rt}) \land \text{sqn } \text{rt } \text{ip} \leq \text{rsn}"
  
  by \text{metis domI option.sel}

show ?thesis
  
  by \text{inv_cterms}

  \text{clarsimp split: split_if_asm option.split_asm elim!: sqninv indests}+

qed

Proposition 7.6

**Lemma sqns_increase:**

\[ \text{paodv } i \models \text{onl } \Gamma_{AODV} (\lambda((\xi, \_), \_, (\xi', \_)). \forall \text{ip}. \text{sqn } \text{rt } \text{ip} \leq \text{sqn } \text{rt' } \text{ip}) \]

proof -
{ fix \xi :: state
  assume *
  "\forall ip \in \text{dom}(\text{dests } \xi). \text{ip} \in \text{kD}(\text{rt } \xi) \land \text{sqn } \text{rt } \text{ip} \leq \text{the (dests } \text{ip})"

  have "\forall ip. \text{sqn } \text{rt } \text{ip} \leq \text{sqn (invalidate } \text{rt } \text{dests}) \text{ip}"

  proof

  fix ip

  from * have "\forall ip \in \text{dom}(\text{dests } \xi) \land \text{sqn } \text{rt } \text{ip} \leq \text{the (dests } \text{ip})" by simp

  thus "\forall ip. \text{sqn } \text{rt } \text{ip} \leq \text{sqn (invalidate } \text{rt } \text{dests}) \text{ip}"

  by \text{metis domI invalidate_sqn option.sel}

  qed

} note solve_invalidate = this

show ?thesis
  
  by \text{inv_cterms}

  \text{clarsimp split: split_if_asm option.split_asm elim!: sqninv indests}+

qed

Proposition 7.7

**Lemma ip_constant:**

\[ \text{paodv } i \models \text{onl } \Gamma_{AODV} (\lambda((\xi, \_), \_, \xi'). \text{ip } \xi = i) \]

by \text{inv_cterms simp add: } \sigma_{AODV}\_def

Proposition 7.8

**Lemma sender_ip_valid':**

\[ \text{paodv } i \models \text{onl } \Gamma_{AODV} (\lambda((\xi, \_), a, \_). \text{anycast } (\lambda m. \text{not}_\text{Pkt } m \Rightarrow \text{msg_sender } m = \text{ip } \xi) a) \]

by \text{inv_cterms}

**Lemma sender_ip_valid:**

\[ \text{paodv } i \models \text{onl } \Gamma_{AODV} (\lambda((\xi, \_), a, \_). \text{anycast } (\lambda m. \text{not}_\text{Pkt } m \Rightarrow \text{msg_sender } m = \text{i}) a) \]

by \text{rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid']}
(auto dest!: onlD onl1D)

lemma received_msg_inv:
  "paodv i |= (recvmsg P |->) onl \( \Gamma_{AODV} (\lambda(\xi, 1). \ l \in \{PAodv\=-:1\} |-> P \ (\text{msg} \ \xi))\)"
  by (inv_cterms)

Proposition 7.9

lemma sip_not_ip':
  "paodv i |= (recvmsg (\lambda m. not_Pkt m |-> msg_sender m |\neq i) |->) onl \( \Gamma_{AODV} (\lambda(\xi, \_). \ \text{sip} \ \xi \neq \text{ip} \ \xi)\)"
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]
  onl_invariant_sterms [OF aodv_wf ip_constant [THEN invariant_restrict_inD]]
  simp add: clear_locals_sip_not_ip' clarsimp+ 

lemma sip_not_ip:
  "paodv i |= (recvmsg (\lambda m. not_Pkt m |-> msg_sender m |\neq i) |->) onl \( \Gamma_{AODV} (\lambda(\xi, \_). \ \text{sip} \ \xi \neq i)\)"
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]
  onl_invariant_sterms [OF aodv_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:
  "paodv i |= onl \( \Gamma_{AODV} (\lambda(\xi, \_). \ \forall \text{ip} \in kD (rt \ \xi). \ \text{the} \ (\text{dhops} (\text{rt} \ \xi) \ \text{ip}) \geq 1)\)"
  by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]) auto

lemma rreq_dip_in_vD_dip_eq_ip:
  "paodv i |= onl \( \Gamma_{AODV} (\lambda(\xi, 1). \ (l \in \{PRreq\=-:16..PRreq\=-:18\} |-> \ \text{dip} \ \xi \in vD(rt \ \xi)) \land (l \in \{PRreq\=-:5, PRreq\=-:6\} |-> \ \text{dip} \ \xi = \text{ip} \ \xi) \land (l \in \{PRreq\=-:15..PRreq\=-:18\} |-> \ \text{dip} \ \xi \neq \text{ip} \ \xi))\)"
  proof (inv_cterms, elim conjE)
  fix \ l \ \xi \ pp \ pp' 
  assume "("\(\xi, pp\) \in \text{reachable (paodv i) TT}"
  and "\{PRreq\=-:17\}[\xi. \ \text{rt} := \text{the} \ (\text{addpreRT} (\text{rt} \ \xi) \ (\text{oip} \ \xi) \ \text{(the} \ \text{(nhop} \ (\text{rt} \ \xi) \ (\text{dip} \ \xi))))]| pp'
  \in \ \text{sterms} \ \Gamma_{AODV} pp" 
  and "l = PRreq\=-:17" 
  and "\text{dip} \ \xi \in vD (rt \ \xi)"
  from this(1-3) have "\text{oip} \ \xi \in kD (rt \ \xi)"
  by (auto dest: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined, where l="PRreq\=-:17"]
  with \(\text{dip} \ \xi \in vD (rt \ \xi)\)
  show "\text{dip} \ \xi \in vD (the (\text{addpreRT} (rt \ \xi) (\text{oip} \ \xi) \ \text{(the} \ \text{(nhop} (rt \ \xi) \ (\text{dip} \ \xi)))))" by simp
 qed

Proposition 7.11

lemma anycast_msg_zhops:
  "\(\forall \text{reqid dip dsn dsK oip osn sip}.
  paodv i |=_A onl1 \( \Gamma_{AODV} (\lambda(\_ , a, \_). \ \text{anycast msg_zhops} a)\)"
  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 \ \xi \ a \ pp \ pp' \ pp'' 
  assume "("\(\xi, pp\) \in \text{reachable (paodv i) TT}"
  and "\{PRreq\=-:18\}\text{unicast}(\lambda(\xi. \ \text{the} \ \text{(nhop} (rt \ \xi) \ (\text{oip} \ \xi)),
  \lambda(\xi. \ \text{Rrep} \ (\text{the} \ \text{(dhops} (\text{rt} \ \xi) \ (\text{dip} \ \xi)))) \ (\text{dip} \ \xi) \ \text{(sqn} (rt \ \xi) \ (\text{dip} \ \xi)) \ (\text{oip} \ \xi) \ (\text{ip} \ \xi)).
  pp' \ > \ pp' \ \in \ \text{sterms} \ \Gamma_{AODV} pp"
  and "l = PRreq\=-:18"
  and "a = \text{unicast} \ (\text{the} \ \text{(nhop} (rt \ \xi) \ (\text{oip} \ \xi))))
  \text{Rrep} \ (\text{the} \ \text{(dhops} (rt \ \xi) \ (\text{dip} \ \xi)))) \ (\text{dip} \ \xi) \ \text{(sqn} (rt \ \xi) \ (\text{dip} \ \xi)) \ (\text{oip} \ \xi) \ (\text{ip} \ \xi))"
  and "\forall \text{ip} \in kD (rt \ \xi). \ \text{Suc} \ 0 \ \text{\leq the} \ \text{(dhops} (rt \ \xi) \ \text{ip})"
  and "\text{dip} \ \xi \in vD (rt \ \xi)"
  from \(\text{dip} \ \xi \in vD (rt \ \xi)\) have "\text{dip} \ \xi \in kD (rt \ \xi)"
  by (rule vD_iD_gives_kD(1))
  with * have "\text{Suc} \ 0 \ \text{\leq the} \ \text{(dhops} (rt \ \xi) \ (\text{dip} \ \xi))" ..
thus "0 < the (dhops (rt ξ) (dip ξ))" by simp

qed

lemma hop_count_zero_oip_dip_sip:
"paodv i |=" (recvmsg msg_zhops \(\rightarrow\) onl \(\Gamma_{AODV}\) \(\lambda(\xi, 1).\)
\((1 \in \{PAodv-:4..PAodv-:5\} \cup \{PRreq-n:n. True\} \rightarrow \langle hops \xi = 0 \rightarrow oip \xi = sip \xi \rangle)\)
\&
\((\langle l \in \{PAodv-:6..PAodv-:7\} \cup \{PRrep-n:n. True\} \rightarrow \langle hops \xi = 0 \rightarrow dip \xi = sip \xi \rangle\)\)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) auto

lemma osn_rreq:
"paodv i |=" (recvmsg rreq_rrep_sn \(\rightarrow\) onl \(\Gamma_{AODV}\) \(\lambda(\xi, l).\)
\(l \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-n:n. True\} \rightarrow 1 \leq osn \xi\)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma osn_rreq':
"paodv i |=" (recvmsg (\(\lambda m. rreq_rrep_sn m \land msg_zhops m\)) \(\rightarrow\) onl \(\Gamma_{AODV}\) \(\lambda(\xi, l).\)
\(l \in \{PAodv-:6, PAodv-:7\} \cup \{PRrep-n:n. True\} \rightarrow 1 \leq osn \xi\)"
proof (rule invariant_weakenE [OF osn_rreq])
  fix a
  assume "recvmsg (\(\lambda m. rreq_rrep_sn m \land msg_zhops m\)) a"
  thus "recvmsg rreq_rrep_sn a"
  by (cases a) simp_all
qed

lemma dsn_rrep:
"paodv i |=" (recvmsg rreq_rrep_sn \(\rightarrow\) onl \(\Gamma_{AODV}\) \(\lambda(\xi, l).\)
\(l \in \{PAodv-:6, PAodv-:7\} \cup \{PRrep-n:n. True\} \rightarrow 1 \leq dsn \xi\)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma dsn_rrep':
"paodv i |=" (recvmsg (\(\lambda m. rreq_rrep_sn m \land msg_zhops m\)) \(\rightarrow\) onl \(\Gamma_{AODV}\) \(\lambda(\xi, l).\)
\(l \in \{PAodv-:6, PAodv-:7\} \cup \{PRrep-n:n. True\} \rightarrow 1 \leq dsn \xi\)"
proof (rule invariant_weakenE [OF dsn_rrep])
  fix a
  assume "recvmsg (\(\lambda m. rreq_rrep_sn m \land msg_zhops m\)) a"
  thus "recvmsg rreq_rrep_sn a"
  by (cases a) simp_all
qed

lemma hop_count_zero_oip_dip_sip':
"paodv i |=" (recvmsg (\(\lambda m. rreq_rrep_sn m \land msg_zhops m\)) \(\rightarrow\) onl \(\Gamma_{AODV}\) \(\lambda(\xi, l).\)
\(l \in \{PAodv-:4..PAodv-:5\} \cup \{PRreq-n:n. True\} \rightarrow \langle hops \xi = 0 \rightarrow oip \xi = sip \xi \rangle\)
\&
\((\langle l \in \{PAodv-:6..PAodv-:7\} \cup \{PRrep-n:n. True\} \rightarrow \langle hops \xi = 0 \rightarrow dip \xi = sip \xi \rangle\)\)"
proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])
  fix a
  assume "recvmsg (\(\lambda m. rreq_rrep_sn m \land msg_zhops m\)) a"
  thus "recvmsg msg_zhops a"
  by (cases a) simp_all
qed

Proposition 7.12

lemma zero_seq_unk_hops_one':
"paodv i |=" (recvmsg (\(\lambda m. rreq_rrep_sn m \land msg_zhops m\)) \(\rightarrow\) onl \(\Gamma_{AODV}\) \(\lambda(\xi, l).\)
\(\forall dip \in kD(rt \xi). \langle sqn (rt \xi) dip = 0 \rightarrow sqnf (rt \xi) dip = unk\rangle\)
\&
\((sqnf (rt \xi) dip = unk \rightarrow the (dhops (rt \xi) dip) = 1)\)
\&
\((the (dhops (rt \xi) dip) = 1 \rightarrow the (nhop (rt \xi) dip) = dip)\)"
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 sqn_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 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. 

132
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]: "\(\forall \text{ sip rt. sqn (update rt sip (0, unk, val, Suc 0, sip, {})) sip) = Suc 0\) →
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_TT [OF aodv_wf addprefRT_welldefined]
onl_invariant_sterms [OF aodv_wf hop_count_zero_oip_dip_sip']
seq_step_invariant_sterms_TT [OF sqns_increase aodv_wf aodv_trans]
onl_invariant_sterms [OF aodv_wf osn_rreq']
onl_invariant_sterms [OF aodv_wf dsn_rrep']) clarsimp+
qed

lemma zero_seq_unk_hops_one:
"paodv i |= (recvmsg (λ m. rreq_rrep_sn m ∧ msg_zhops m) → onl Γ_AODV (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:
"paodv i |= (recvmsg rreq_rrep_sn → onl Γ_AODV (λ (ξ, l).
∀ dip∈kD(rt ξ). π₃ (the (rt ξ dip)) = unk ∨ Suc 0 ≤ sqn rt 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)+
}

{ fix dip sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2
  assume allkd: "∀ dip∈kD(rt). π₃ (the (rt dip)) = unk ∨ Suc 0 ≤ sqn dip"
  and **: "dsk1 = unk ∨ Suc 0 ≤ dsn2"
  have "∀ dip∈kD(rt). π₃ (the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1, pre1) dip)) = 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
}

{ 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∈dom(dests)" [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 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 |=_(A (recvmsg rreq_rrep_sn → onl Γ AODV (λ(_, a, _). anycast rreq_rrep_sn a))"
proof -
  have sqnf_kno: "paodv i |= onl Γ AODV (λ(ξ, 1).
  (1 ∈ {PRreq-:16..PRreq-:18} → sqn (rt ξ) (dip ξ) = kno))"
  proof
    inv_cterms
    fix l ξ l' pp p'
    assume "("ξ, pp) ∈ reachable (paodv i) TT"
and \( \{\text{PRreq-:2}\} \lambda \xi. \xi(\text{rt} := \text{update (rt} \xi) \{\text{osn} \xi, \text{kno}, \text{val}, \text{Suc (hops} \xi), \text{sip} \xi, \{\})\} p' \in \text{sterms} \Gamma_{\text{AODV}} \text{ pp} \) and \( \{1' = \text{PRreq-:3}\} \)

show \( \text{osn} \xi < \text{sqn (update (rt} \xi) \{\text{osn} \xi, \text{kno}, \text{val}, \text{Suc (hops} \xi), \text{sip} \xi, \{\})\} \{\text{oip} \xi\} \text{ (osn} \xi, \text{kno}, \text{val}, \text{Suc (hops} \xi), \text{sip} \xi, \{\})\) \{\text{oip} \xi\} = \text{osn} \xi \leq \text{Suc (hops} \xi) \)

\( \text{unfolding update_def by (clarsimp split: option.split)} \)

\( \text{by (metis linorder_neqE_nat not_less)} \)

\(\text{qed}\)

have \(\text{rrep_prrep: "paodv i} \models \text{onl} \Gamma_{\text{AODV}} (\lambda(\xi, 1). \text{ (1} \in \{\text{PRrep-:2..PRrep-:7}\} \rightarrow (\text{dip} \xi \in \text{kD(rt} \xi) \)

\( \wedge \text{sqn (rt} \xi) (\text{dip} \xi) = \text{dsn} \xi \)

\( \wedge \text{the (dhops (rt} \xi) (\text{dip} \xi)) = \text{Suc (hops} \xi) \)

\( \wedge \text{the (flag (rt} \xi) (\text{dip} \xi)) = \text{val} \)

\( \wedge \text{the (nhop (rt} \xi) (\text{dip} \xi)) \in \text{kD (rt} \xi)))" \)

by \(\text{(inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rrep_1_update_changes] onl_invariant_sterms [OF aodv_wf sip_in_kD])}\)

show \(?\text{thesis}\)


\(\text{qed}\)

Proposition 7.15

**Lemma rerr_invalid_any_step_invariant:**

\(\text{"paodv i} \models \text{onl} \Gamma_{\text{AODV}} (\lambda(\xi, 1). \text{ a, } \_). \text{anycast (rerr_invalid (rt} \xi)) a")\)

**Proof -**

have \(\text{dests_inv: "paodv i} \models \text{onl} \Gamma_{\text{AODV}} (\lambda(\xi, 1). \text{ (1} \in \{\text{PAodv-:15, PPkt-:7, PRreq-:9, PRreq-:21, PRrep-:10, PRerr-:1}\}

\( \rightarrow (\forall \text{ip}\in\text{dom(dests} \xi)\). \text{ip}\in\text{iD(rt} \xi))) \)

\( \wedge (1 \in \{\text{PAodv-:16..PAodv-:19}\}

\cup \{\text{PPkt-:8..PPkt-:11}\}

\cup \{\text{PRreq-:10..PRreq-:13}\}

\cup \{\text{PRreq-:22..PRreq-:25}\}

\cup \{\text{PRrep-:11..PRrep-:14}\}

\cup \{\text{PRerr-:2..PRerr-:5}\} \rightarrow (\forall \text{ip}\in\text{dom(dests} \xi)\). \text{ip}\in\text{iD(rt} \xi))

\wedge (\text{the (dests} \xi \text{ ip}) = \text{sqn (rt} \xi \text{ ip})) \)

\( \wedge (1 = \text{PPkt-:14} \rightarrow \text{dip} \xi \in\text{iD(rt} \xi)))" \)

by \(\text{inv_cterms (clarsimp split: split_if_asm option.split_asm simp: domIff)+}\)

show \(\text{?thesis}\)

by \(\text{(inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_inv])}\)

\(\text{qed}\)

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 \((\text{rip}, \text{rsn})\) is assigned to the variable \text{dests} of type \(\text{ip} \rightarrow \text{sqn}\), or to the first argument of the function \text{rerr}, this set is a partial function, i.e., there is at most one entry \((\text{rip}, \text{rsn})\) for each destination \text{rip}: guaranteed by type.

**Lemma dests_vD_inc_sqn:**

\(\text{"paodv i} \models \)

\(\text{135}\)
onl $\Gamma_{AODV}(\lambda(\xi, 1). (1 \in \{P_{aodv}=15, P_{ppkt}=7, P_{rreq}=9, P_{rreq}=21, P_{rreq}=10\})$
\[\rightarrow (\forall ip \in \text{dom}(dests) . \text{ip} \in \text{V}(rt) \wedge \text{the} (dests ip) = \text{inc} (\text{sqn (rt)} \xi ip)) \]
\[\wedge (1 = P_{rerr}=1) \rightarrow (\forall ip \in \text{dom}(dests) . \text{ip} \in \text{V}(rt) \wedge \text{the} (dests ip) > \text{sqn (rt)} \xi ip))"\]
by inv_terms (clarsimp split: split_if_asm option.split_asm)+

Proposition 7.27

lemma route_tables_fresher:
"paodv i \models (recvmsg rreq_rrep_sn \rightarrow) onll $\Gamma_{AODV}(\lambda((\xi, \_), \_ (\xi', \_)).
\forall dip \in kD(rt) . \text{rt} (\xi \sqsubseteq dip \text{rt} \xi')"\nproof (inv_terms inv add:
\[\text{onl}_1 \text{invariant_terms} \{\text{OF} \ aodv_wf dests_V_inc_sqn \{\text{THEN} \ \text{invariant_restrict_inD}\}\]
\[\text{onl}_1 \text{invariant_terms} \{\text{OF} \ aodv_wf hop_count_positive \{\text{THEN} \ \text{invariant_restrict_inD}\}\]
\[\text{onl}_1 \text{invariant_terms} \{\text{OF} \ aodv_wf osn_rreq\}
\[\text{onl}_1 \text{invariant_terms} \{\text{OF} \ aodv_wf dsn_rrep\}
\[\text{onl}_1 \text{invariant_terms} \{\text{OF} \ aodv_wf addpreRT_welldefined \{\text{THEN} \ \text{invariant_restrict_inD}\}\}
fix $\xi$ pp p'
assume "((\xi, pp) \in \text{reachable (paodv i) (recvmsg rreq_rrep_sn)}"
and "\{P_{rreq}=2\}\lambda(\xi, (rt := \text{update (rt)} (\text{oip} \xi) \text{(osn} \xi, \text{kno, val, Suc (hops} \xi, \text{sip} \xi, \{\}))\]
p' \in \text{stems} \Gamma_{AODV} pp"
and "\text{Suc 0} \leq \text{osn} \xi"
and "\forall ip \in kD (rt \xi). \text{Suc 0} \leq \text{the} (\text{dhops (rt)} \xi ip)"
show "\forall ip \in kD (rt \xi). \text{rt} \xi \sqsubseteq dip \text{update (rt)} (\text{oip} \xi) \text{(osn} \xi, \text{kno, val, Suc (hops} \xi, \text{sip} \xi, \{))"
proof
fix ip
assume "ip \in kD (rt \xi)"
moreover with * have "\text{1} \leq \text{the} (\text{dhops (rt)} \xi ip)" by simp
moreover from \text{Suc 0} \leq \text{osn} \xi
have "\text{update_arg_wf (osn} \xi, \text{kno, val, Suc (hops} \xi, \text{sip} \xi, \{))" ..
ultimately show "\text{rt} \xi \sqsubseteq dip \text{update (rt)} (\text{oip} \xi) \text{(osn} \xi, \text{kno, val, Suc (hops} \xi, \text{sip} \xi, \{))"
by (rule rt_fresher_update)
qed
next
fix $\xi$ pp p'
assume "((\xi, pp) \in \text{reachable (paodv i) (recvmsg rreq_rrep_sn)}"
and "\{P_{rreq}=2\}\lambda(\xi, (rt := \text{update (rt)} (\text{dip} \xi) \text{(dsn} \xi, \text{kno, val, Suc (hops} \xi, \text{sip} \xi, \{))\]
p' \in \text{stems} \Gamma_{AODV} pp"
and "\text{Suc 0} \leq \text{dsn} \xi"
and "\forall ip \in kD (rt \xi). \text{Suc 0} \leq \text{the} (\text{dhops (rt)} \xi ip)"
show "\forall ip \in kD (rt \xi). \text{rt} \xi \sqsubseteq dip \text{update (rt)} (\text{dip} \xi) (\text{dsn} \xi, \text{kno, val, Suc (hops} \xi, \text{sip} \xi, \{))"
proof
fix ip
assume "ip \in kD (rt \xi)"
moreover with * have "\text{1} \leq \text{the} (\text{dhops (rt)} \xi ip)" by simp
moreover from \text{Suc 0} \leq \text{dsn} \xi
have "\text{update_arg_wf (dsn} \xi, \text{kno, val, Suc (hops} \xi, \text{sip} \xi, \{))" ..
ultimately show "\text{rt} \xi \sqsubseteq dip \text{update (rt)} (\text{dip} \xi) (\text{dsn} \xi, \text{kno, val, Suc (hops} \xi, \text{sip} \xi, \{))"
by (rule rt_fresher_update)
qed
next
end

1.7 The quality increases predicate

theory A_Quality_Improves
imports A_Aodv_Predicates A_Fresher
begin

definition quality_increases :: "\text{state} \Rightarrow \text{state} \Rightarrow \text{bool}"
where "quality_increases \xi \xi' \equiv (\forall dip \in kD(rt \xi). dip \in kD(rt \xi') \wedge rt \xi \sqsubseteq dip \text{rt} \xi')"
\[\wedge (\forall dip. \text{sqn} \text{rt} \xi dip \leq \text{sqn (rt} \xi') \text{dip})"

lemma quality_increasesI [intro!]:
end
assumes \( \land \text{dip. dip} \in kD(rt \xi) \Longrightarrow \text{dip} \in kD(rt \xi') \) 
and \( \land \text{dip. dip} \in kD(rt \xi); \text{dip} \in kD(rt \xi') \] \( \Longrightarrow \) \( rt \xi \sqsubseteq \text{dip} \) \( rt \xi' \) 
and \( \land \text{sqn} (rt \xi) \text{dip} \leq \text{sqn} (rt \xi') \text{dip} \)
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 "\( \text{dip} \in kD(rt \xi) \)"
and "\([ \text{dip} \in kD(rt \xi'); rt \xi \sqsubseteq \text{dip} \) \( rt \xi' \); \text{sqn} (rt \xi) \text{dip} \leq \text{sqn} (rt \xi') \text{dip} \] \( \Longrightarrow \) 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 "\( \text{ip} \in kD(rt \xi) \)"

shows "rt \( \xi \sqsubseteq \text{ip} \) \( \xi' \)"

using assms by auto

lemma quality_increases_sqnE [elim]:
fixes dip
assumes "quality_increases \( \xi \xi' \)"
and "\( \text{sqn} (rt \xi) \text{dip} \leq \text{sqn} (rt \xi') \text{dip} \) \( \Longrightarrow \) 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 "\( \text{dip} \in kD(rt (\sigma \ nhip)) \)"

shows "rt \( (\sigma \ i) \sqsubseteq \text{dip} \) \( \sigma' \ nhip \)"

proof -
from qinc have "rt \( (\sigma \ nhip) \sqsubseteq \text{dip} \) rt \( (\sigma' \ nhip) \)" using \( \text{dip} \in kD(rt (\sigma \ nhip)) \)
by auto
with \( rt (\sigma \ i) \sqsubseteq \text{dip} \) rt \( (\sigma' \ nhip) \) show ?thesis ..
qed

lemma kD_quality_increases [elim]:
assumes "\( \text{i} \in kD(rt \xi) \)"
and "quality_increases \( \xi \xi' \)"

shows "\( \text{i} \in kD(rt \xi') \)"

using assms by auto

lemma kD_nsqn_quality_increases [elim]:
assumes "\( \text{i} \in kD(rt \xi) \)"
and "quality_increases \( \xi \xi' \)"

shows "\( \text{i} \in kD(rt \xi') \) \( \text{nsqn} \) (rt \( \xi \)) \text{i} \leq \text{nsqn} \) (rt \( \xi' \)) \text{i}"

proof -
from assms have "\( \text{i} \in kD(rt \xi') \) ..
moreover with assms have "rt \( \xi \sqsubseteq_1 \) rt \( \xi' \)" by auto
ultimately have "\( \text{nsqn} \) (rt \( \xi \)) \text{i} \leq \text{nsqn} \) (rt \( \xi' \)) \text{i}"
using \( \text{i} \in kD(rt \xi) \) by - (erule(2) rt_fresher_imp_nsqn_le)
with \( \text{i} \in kD(rt \xi') \) show ?thesis ..
qed

lemma nsqn_quality_increases [elim]:
assumes "\( \text{i} \in kD(rt \xi) \)"
and "quality_increases \( \xi \xi' \)"

...
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 "s ≤ nsqn (rt (σ sip)) ip ≤ nsqn (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 \( \langle \text{ip} \in kD(\text{rt} (\sigma \text{ sip})) \rangle \) and \( \langle \text{quality\_increases} (\sigma \text{ sip}) (\sigma' \text{ sip}) \rangle \)

have "\( \ldots \leq \text{nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ ip} \)" ...
finally have "\( \text{sn} < \text{nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ ip} \)".
thus ?thesis by simp

next
assume "\( \text{sn} = \text{nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ ip} \)"
with \( \langle \text{ip} \in kD(\text{rt} (\sigma \text{ sip})) \rangle \) and \( \langle \text{quality\_increases} (\sigma \text{ sip}) (\sigma' \text{ sip}) \rangle \)

have "\( \text{sn} < \text{nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ ip} \)
\( \lor (\text{sn} = \text{nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ ip} \land (\text{the} (\text{dhops} (\text{rt} (\sigma' \text{ sip})) \text{ ip})) \leq (\text{dhops} (\text{rt} (\sigma \text{ sip})) \text{ ip}))" ..

hence "\( \text{sn} < \text{nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ ip} \)
\( \lor (\text{nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ ip} = \text{sn} \land (\text{the} (\text{dhops} (\text{rt} (\sigma' \text{ sip})) \text{ ip}) \leq \text{hops}
\lor (\text{the} (\text{flag} (\text{rt} (\sigma' \text{ sip})) \text{ ip}) = \text{inv}))""

proof
assume "\( \text{sn} < \text{nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ ip} \)" thus ?thesis ..
next
assume "\( \text{sn} = \text{nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ ip} \)
\( \land (\text{the} (\text{dhops} (\text{rt} (\sigma \text{ sip})) \text{ ip}) \geq (\text{dhops} (\text{rt} (\sigma' \text{ sip})) \text{ ip}))""

hence "the (\text{dhops} (\text{rt} (\sigma' \text{ sip})) \text{ ip}) \leq \text{hops} by simp
with \( \langle \text{sn} = \text{nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ ip} \rangle \) have "\( \text{the} (\text{dhops} (\text{rt} (\sigma \text{ sip})) \text{ ip}) \leq \text{hops} \)"
by simp
thus ?thesis

proof
assume "\( \text{the} (\text{flag} (\text{rt} (\sigma \text{ sip})) \text{ ip}) = \text{inv} \)"
with \( \langle \text{ip} \in kD(\text{rt} (\sigma \text{ sip})) \rangle \) have "\( \text{the} (\text{dhops} (\text{rt} (\sigma \text{ sip})) \text{ ip}) \leq \text{hops} \)
(\text{the} (\text{flag} (\text{rt} (\sigma \text{ sip})) \text{ ip}) = \text{inv})""

qed
lemma quality_increases_rreq_rrep_props:
  fixes sn ip hops sip
assumes "\( \forall j. \text{quality_increases} (\sigma j) (\sigma' j) \)"
and "1 \( \leq \) sn"
and ": "ip \( \in \) kD(rt (\sigma sip)) \( \land \) sn \( \leq \) nsqn (rt (\sigma sip)) ip
\( \land \) (nsqn (rt (\sigma sip)) ip = sn
\( \rightarrow \) (the (dhops (rt (\sigma sip)) ip) \( \leq \) hops
\( \lor \) the (flag (rt (\sigma sip)) ip) = inv))"
shows "ip \( \in \) kD(rt (\sigma sip)) \( \land \) sn \( \leq \) nsqn (rt (\sigma sip)) ip
\( \land \) (nsqn (rt (\sigma sip)) ip = sn
\( \rightarrow \) (the (dhops (rt (\sigma sip)) ip) \( \leq \) hops
\( \lor \) 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 "\( \forall j. j \neq i \rightarrow \text{quality_increases} (\sigma j) (\sigma' j) \)"
  and "rt (\sigma' i) = 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 \text{state}) \Rightarrow \text{msg}) \Rightarrow \text{bool}"
where "msg_fresh \sigma m \equiv \text{case } m \text{ of } Rreq \text{ hopsc } oipc osnc sipc \Rightarrow \text{ osnc } \geq 1 \land (sipc \neq oipc \rightarrow oipc \in kD(rt (\sigma sipc)) \land nsqn (rt (\sigma sipc)) oipc \geq osnc \land (nsqn (rt (\sigma sipc)) oipc = osnc \rightarrow (hopsc \geq \text{the (dhops (rt (\sigma sipc)) oipc)} \lor \text{the (flag (rt (\sigma sipc)) oipc) = inv})))"
| Rrep hopsc dipc dsnc sipc \Rightarrow \text{ dsnc } \geq 1 \land (sipc \neq dipc \rightarrow dipc \in kD(rt (\sigma sipc)) \land nsqn (rt (\sigma sipc)) dipc \geq dsnc \land (nsqn (rt (\sigma sipc)) dipc = dsnc \rightarrow (hopsc \geq \text{the (dhops (rt (\sigma sipc)) dipc)} \lor \text{the (flag (rt (\sigma sipc)) dipc) = inv})))"
| Rerr destsc sipc \Rightarrow (\forall ripc \in \text{dom(destsc)}. (ripc \in kD(rt (\sigma sipc)) \land \text{the (destsc ripc) - 1 } \leq \text{nsqn (rt (\sigma sipc)) ripc}))"
| _ \Rightarrow True"

lemma msg_fresh [simp]:
"\( \forall \text{dip dip dsn dsk oip osn sip}. \) msg_fresh \sigma (Rreq hops dip dsn dsk oip osn sip) =
  (osn \geq 1 \land (sipc \neq oipc \rightarrow oipc \in kD(rt (\sigma sipc)) \land nsqn (rt (\sigma sipc)) oipc \geq osn \land (nsqn (rt (\sigma sipc)) oipc = osn \rightarrow (hopsc \geq \text{the (dhops (rt (\sigma sipc)) oipc)} \lor \text{the (flag (rt (\sigma sipc)) oipc) = inv}))))"

"\( \forall \text{dip dip dsn oip sip}. \) msg_fresh \sigma (Rreq hops dip dsn oip sip) =
  (dsn \geq 1 \land (sipc \neq dipc \rightarrow dipc \in kD(rt (\sigma sipc)) \land nsqn (rt (\sigma sipc)) dipc \geq dsn \land (nsqn (rt (\sigma sipc)) dipc = dsn \rightarrow (hopsc \geq \text{the (dhops (rt (\sigma sipc)) dipc)} \lor \text{the (flag (rt (\sigma sipc)) dipc) = inv}))))"

"\( \forall \text{dests sip}. \) msg_fresh \sigma (Rerr dests sip) =
  (\forall ripc \in \text{dom(dests)}. (ripc \in kD(rt (\sigma sipc)) \land \text{the (dests ripc) - 1 } \leq \text{nsqn (rt (\sigma sipc)) ripc}))"

"\( \forall \text{d dip}. \) msg_fresh \sigma (Newpkt d dip) = True"
"\( \forall \text{d dip sip}. \) msg_fresh \sigma (Pkt d dip sip) = True"

unfolding msg_fresh_def by simp_all

lemma msg_fresh_inc_sn [simp, elim]:
"\( \forall \text{m}. \) msg_fresh \sigma m \( \Rightarrow \) rreq_rrep_sn m"
by (cases m) simp_all
lemma recv_msg_fresh_inc_sn [simp, elim]:
"orecvmsg (msg_fresh) σ m ⇒ recvmsg rreq_rrep_sn m"
by (cases m) simp_all

lemma rreq_nsqn_is_fresh [simp]:
  fixes σ msg hops dip dsn oip osn sip
  assumes "rreq_rrep_fresh (rt (σ sip)) (Rreq hops dip dsn oip osn sip)"
  and "rreq_rrep_sn (Rreq hops dip dsn oip osn sip)"
  shows "msg_fresh σ (Rreq hops dip dsn 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" ..
      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 osn sip
  assumes "rreq_rrep_fresh (rt (σ sip)) (Rrep hops dip dsn oip osn sip)"
  and "rreq_rrep_sn (Rrep hops dip dsn oip osn sip)"
  shows "msg_fresh σ (Rrep hops dip dsn oip osn sip)"
  (is "msg_fresh σ ?msg")
proof -
let \(?rt = \text{"rt (}\sigma\text{ sip)}"
from \text{assms have } \text{"sip \neq dip }\rightarrow\text{ dip} \in \text{kD(?rt)} \land \text{sqn ?rt dip = dsn} \land \text{the (flag ?rt dip) = val}"
by simp
hence \text{"sip \neq dip }\rightarrow\text{ dip} \in \text{kD(?rt)} \land \text{nsqn ?rt dip }\geq\text{ dsn}"
by clarsimp
with \text{assms show } \text{"msg\_fresh }\sigma\text{ ?msg}"
by clarsimp
qed

lemma \text{rerr\_nsqn\_is\_fresh [simp]}:
fixes \sigma \text{ msg dests sip}
assumes \text{"rerr\_invalid (rt (}\sigma\text{ sip)) (Rerr dests sip)"
shows \text{"msg\_fresh }\sigma\text{ (Rerr dests sip)"
(is \text{"msg\_fresh }\sigma\text{ ?msg})
proof -
let \(?rt = \text{"rt (}\sigma\text{ sip)}"
from \text{assms have } \text{*: }\text{ (}\forall \text{rip }\in \text{dom(dests). (rip} \in \text{iD(rt (}\sigma\text{ sip})) \land \text{the (dests rip) = sqn (rt (}\sigma\text{ sip)) rip)}"
by clarsimp
have \text{"(}\forall \text{rip }\in \text{dom(dests). (rip} \in \text{kD(rt (}\sigma\text{ sip})) \land \text{the (dests rip) }-\text{ 1 }\leq\text{ nsqn (rt (}\sigma\text{ sip)) rip)}"
proof
fix rip
assume \text{"rip }\in\text{ dom dests}"
with \text{* have } \text{"rip} \in \text{iD(rt (}\sigma\text{ sip))"
and \text{"the (dests rip) = sqn (rt (}\sigma\text{ sip)) rip}"
by auto
from this(2) have \text{"the (dests rip) }-\text{ 1 }\leq\text{ nsqn (rt (}\sigma\text{ sip)) rip}"
finally have \text{"the (dests rip) }-\text{ 1 }\leq\text{ nsqn (rt (}\sigma\text{ sip)) rip}"

with \text{\langle rip }\in \text{iD(rt (}\sigma\text{ sip))\rangle}
show \text{"rip} \in \text{kD(rt (}\sigma\text{ sip)) \land the (dests rip) }-\text{ 1 }\leq\text{ nsqn (rt (}\sigma\text{ sip)) rip}"
by clarsimp
qed
thus \text{"msg\_fresh }\sigma\text{ ?msg"}
by simp
qed

lemma \text{quality\_increases\_msg\_fresh [elim]}:
assumes \text{qinc: }\text{"}\forall j. \text{quality\_increases (}\sigma\text{ j) (}\sigma\prime\text{ j)"
and \text{"msg\_fresh }\sigma\text{ m"
shows \text{"msg\_fresh }\sigma\prime\text{ m"
using \text{assms(2)}
proof \text{(cases m)}
fix hops rreqid dip dsn dsk oip osn sip
assume \text{[simp]: } m = \text{Rreq hops dip dsn dsk oip osn sip}"
and \text{"msg\_fresh }\sigma\text{ m"
then have \text{"osn }\geq\text{ 1" and \text{"sip = oip }\lor (oip} \in \text{kD(rt (}\sigma\text{ sip)) \land osn }\leq\text{ nsqn (rt (}\sigma\text{ sip)) oip} \land (\text{nsqn (rt (}\sigma\text{ sip)) oip }=\text{ osn} \lor \text{the (dhops (rt (}\sigma\text{ sip)) oip) }\leq\text{ hops} \lor \text{the (flag (rt (}\sigma\text{ sip)) oip) = inv)))"
by auto
from this(2) show \text{?thesis}
proof
assume \text{"sip = oip" with }\text{osn }\geq\text{ 1} show \text{?thesis by simp
next
assume \text{"oip} \in \text{kD(rt (}\sigma\text{ sip)) \land osn }\leq\text{ nsqn (rt (}\sigma\text{ sip)) oip} \land (\text{nsqn (rt (}\sigma\text{ sip)) oip }=\text{ osn} \lor \text{the (dhops (rt (}\sigma\text{ sip)) oip) }\leq\text{ hops} \lor \text{the (flag (rt (}\sigma\text{ sip)) oip) = inv)))"
moreover from \text{qinc have }\text{"quality\_increases (}\sigma\text{ sip) (}\sigma\prime\text{ sip)"
ultimately have \text{"oip} \in \text{kD(rt (}\sigma\prime\text{ sip)) \land osn }\leq\text{ nsqn (rt (}\sigma\prime\text{ 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))
    ∧ (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))
    ∧ (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"
    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 - (erule(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

1.8 The ‘open’ AODV model

definition A_OAodv :: "((ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp)) set" where "A_OAodv = {λi. aodv_init i, Γ_aodv PAodv}"
1.9 Global invariant proofs over sequential processes
lemma orceived_msg_inv:
assumes other: "∀ σ σ' m. [ P σ m; other Q {i} σ σ' ] ⇒ P σ' m"
and local: "∀ σ m. P σ m ⇒ P (σ(i := σ i(msg := m))) m"
shows "opaodv i |= (otherwith Q {i} (orecmsg P), other Q {i} →)
onl ΓAODV (λ (σ, l). l ∈ {PAodv-:1} → P σ (msg (σ i)))"

proof
(thm inv_cterms, intro impI)
fix σ σ l
assume "l = PAodv-:1 → P σ (msg (σ i))"
and "l = PAodv-:1"
and "other Q {i} σ σ'"
from this(1-2) have "P σ (msg (σ i))" ..
hence "P σ' (msg (σ i))" using ⟨other Q {i} σ σ'⟩
by (rule other)
moreover from ⟨other Q {i} σ σ'⟩ have "σ' i = σ i" ..
ultimately show "P σ' (msg (σ' i))" by simp

next
fix σ σ' msg
assume "otherwith Q {i} (orecmsg 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

(Equivalent to) Proposition 7.27

lemma local_quality_increases:
"paodv i |=A (recmsg req_rrep_sn →) onll ΓAODV (λ((ξ, _), (_, _, (ξ', _))). quality_increases (ξ ξ'))"

proof
(rule step_invariantI)
fix s a s'
assume sr: "s ∈ reachable (paodv i) (recmsg req_rrep_sn)"
and tr: "(s, a, s') ∈ trans (paodv i)"
and rm: "recmsg req_rrep_sn a"
from sr have srTT: "s ∈ reachable (paodv 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_aodv aoadv_trans aoadv_trans, simplified seqll_onll_swap]

lemma oquality_increases:
"opaodv i |=A (otherwith quality_increases {i} (orecmsg (λ_. req_rrep_sn)), other quality_increases {i} →)"
lemmas rreq_rrep_nsqn_fresh_any_step_invariant:
| (rule ostep_invariantI, simp)
| proof: (rule onll_ostep_invariantI, simp)
| proof -
| obtain 1 l' where "l' ∈ labels Γ AODV p" and "l' ∈ labels Γ AODV p'' "
| have tr: "((σ, p), a, (σ', p')) ∈ trans (opaodv i)" by simp
| have "anycast (rreq_rrep_fresh (rt (σ i))) a"
| proof -
| have "opaodv i ⊢ A (act (recvmsg rreq_rrep_sn), other A {i} →)
| onll Γ AODV (λ((ξ, _), a, _). anycast (msg_fresh σ) a)"
| by (rule onll_ostep_invariantI, simp del: act_simp)
| have "opaodv i ⊢ A (act (recvmsg rreq_rrep_sn), other A {i} →)
| onll Γ AODV (seqll i (λ((ξ, _), a, _). anycast (rreq_rrep_fresh (rt ξ)) a))"
| by (rule onll_ostep_invariant_weakenE [OF open_seq_step_invariant [OF rreq_rrep_fresh_any_step_invariant initiali_aodv, simplified seqll_onll_swap]) auto
| hence "onll Γ AODV (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_inverse)
| thus ?thesis
| using l ∈ labels Γ AODV p and l' ∈ labels Γ AODV p'' by auto
| qed

moreover have "anycast (rerr_invalid (rt (σ i))) a"
| proof -
| have "opaodv i ⊢ A (act (recvmsg rreq_rrep_sn), other A {i} →)
| onll Γ AODV (seqll i (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a))"
| by (rule onll_ostep_invariant_weakenE [OF open_seq_step_invariant [OF rerr_invalid_any_step_invariant initiali_aodv, simplified seqll_onll_swap]) auto
| hence "onll Γ AODV (seqll i (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a))
| ((σ, p), a, (σ', p'))"
| using or tr recv by - (erule(4) ostep_inverse)
| thus ?thesis
| using l ∈ labels Γ AODV p and l' ∈ labels Γ AODV p'' by auto
| qed

moreover have "anycast rreq_rrep_sn a"
proof -
  from or tr recv
  have "onll Γ_{AODV} (seqll i (λ(_, a, _). ancast 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_aodv
        oadv_trans adv_trans,
        simplified seqll_onll_swap]])
  thus ?thesis
    using ⟨l ∈ labels Γ_{AODV} p⟩ and ⟨l' ∈ labels Γ_{AODV} p'⟩ by auto
  qed

moreover have "anycast (λm. not_Pkt m −→ msg_sender m = i) a"
proof -
  have "opaodv i |−_A (act (recvmsg rreq_rrep_sn), other A {i} →)
    onll Γ_{AODV} (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 Γ_{AODV} p⟩ and ⟨l' ∈ labels Γ_{AODV} 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 Γ_{AODV} (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 |−_A (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onll Γ_{AODV} (λ(σ, 1). 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 Γ_{AODV} (λ((σ, _, _), _, (σ', _)). ∀ j. quality_increases (σ j) (σ' j))"
by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
  "opaodv i |−_A (otherwith quality_increases {i} (orecvmsg msg_fresh),
    other quality_increases {i} →)
    onll Γ_{AODV} (seqll i (λ((ξ, 1), 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: seqll_onll_swap)
lemma rreq_sip:
"opaodv i \vdash \{\text{other with quality increases \{i\} (orecvmsg msg\_fresh),
other quality increases \{i\} \rightarrow}\)
\text{onl} \Gamma_{AODV} (\lambda(\sigma, i)\ldots
(1 \in \{\text{PAodv-:4, PAodv-:5, PRreq-:0, PRreq-:2} \land \text{sip} (\sigma i) \neq \text{opi} (\sigma i))\ldots
\rightarrow \text{opi} (\sigma i) \in kD (rt (\sigma (\text{sip} (\sigma i))))\ldots
\land \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i)) \geq \text{osn} (\sigma i)\ldots
\land (\text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i)) = \text{osn} (\sigma i)\ldots
\rightarrow (\text{dhops} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i))) \geq (\text{hops} (\sigma i))\ldots
\lor (\text{the} (\text{flag} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i)))) = (\text{inv}))\ldots"
(is \_\_ \vdash (\text{?S, \text{?U} \rightarrow \_})")

proof (inv\_cterm inv add: oseq\_step\_invariant\_stems \[\text{OF equality\_increases\_nsqn\_fresh} \aodv\_wf \oaodv\_trans\]
\text{onl\_oinvariant\_stems} \[\text{OF aodv\_wf orecvmsg\_msg\_fresh\_inv}\]
\text{onl\_oinvariant\_stems} \[\text{OF aodv\_wf oosn\_rrep}\]
simp add: seqlsimp
simp del: \text{One\_nat\_def, rule implI})

fix \sigma \sigma' \in l
assume "(\sigma, p) \in \text{oreachable} (opaodv i) \text{?S} \text{?U}"
and "1 \in \text{labels} \Gamma_{AODV} p"
and pre:
"(1 = \text{PAodv-:4} \lor 1 = \text{PAodv-:5} \lor 1 = \text{PRreq-:0} \lor 1 = \text{PRreq-:2} \land \text{sip} (\sigma i) \neq \text{opi} (\sigma i)"
\rightarrow (\text{opi} (\sigma i) \in \text{kD} (rt (\sigma (\text{sip} (\sigma i))))\ldots
\land \text{osn} (\sigma i) \leq \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i))\ldots
\land (\text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i)) = \text{osn} (\sigma i)\ldots
\rightarrow (\text{dhops} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i))) \leq (\text{hops} (\sigma i))\ldots
\lor (\text{the} (\text{flag} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i)))) = (\text{inv}))" and "other quality increases \{i\} \sigma \sigma'"
and hyp: "(1 = \text{PAodv-:4} \lor 1 = \text{PAodv-:5} \lor 1 = \text{PRreq-:0} \lor 1 = \text{PRreq-:2} \land \text{sip} (\sigma i) \neq \text{opi} (\sigma i)"
(is "\text{?labels} \land \text{sip} (\sigma i) \neq \text{opi} (\sigma i)")
from (4) have "\sigma i = \sigma i" ..
with hyp have hyp': "\text{?labels} \land \text{sip} (\sigma i) \neq \text{opi} (\sigma i)" by simp
show "\text{opi} (\sigma i) \in \text{kD} (rt (\sigma (\text{sip} (\sigma i))))\ldots
\land \text{osn} (\sigma i) \leq \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i))\ldots
\land (\text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i)) = \text{osn} (\sigma i)\ldots
\rightarrow (\text{dhops} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i))) \leq (\text{hops} (\sigma i))\ldots
\lor (\text{the} (\text{flag} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\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 "other quality increases \{i\} (\sigma (\text{sip} (\sigma i))) (\sigma' (\text{sip} (\sigma i)))" by (rule otherE (clarlsimp simp: \text{sip} (\sigma i) \neq i))
moreover from (\sigma, p) \in \text{oreachable} (opaodv i) \text{?S} \text{?U} \{l \in \text{labels} \Gamma_{AODV} p\} and hyp
have "1 \leq \text{osn} (\sigma i)"
by (auto dest!: \text{onl\_oinvariant\_weakenD} \[\text{OF oosn\_rreq}\]
simp add: seqlsimp \sigma' = \sigma i)
moreover from \text{sip} (\sigma i) \neq i hyp' and pre
have "\text{opi} (\sigma i) \in \text{kD} (rt (\sigma (\text{sip} (\sigma i))))\ldots
\land \text{osn} (\sigma i) \leq \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i))\ldots
\land (\text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i)) = \text{osn} (\sigma i)\ldots
\rightarrow (\text{dhops} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i))) \leq (\text{hops} (\sigma i))\ldots
\lor (\text{the} (\text{flag} (rt (\sigma (\text{sip} (\sigma i)))) (\text{opi} (\sigma i)))) = (\text{inv}))" by (auto simp: \sigma' = \sigma i)
ultimately show \text{?thesis}
by (rule quality\_increases\_rrep\_rrep\_props)
next
assume "\text{sip} (\sigma i) = i" thus \text{?thesis}
using \sigma' = \sigma i hyp and pre by auto
qed
qed (auto elim!: quality\_increases\_rrep\_rrep\_props')

lemma odsn\_rrep:
"opaodv i \vdash \{\text{other with quality increases \{i\} (orecvmsg msg\_fresh),
other quality increases \{i\} \rightarrow}\)

148
lemma rrep_sip:
"opaodv i \models (otherwith quality_increases \{i\} (orecvmsg msg_fresh),
other quality_increases \{i\} \rightarrow)
onl \Gamma_{AODV} (λ(σ, i).
(1 \in \{PAodv-:6, PAodv-:7, PRrep-:0, PRrep-:1\} \land 
-seql_onl_swap

\begin{proof}
\begin{quote}
(\text{inv}_{\text{terms}} \text{ inv add: oseq_step_invariant_\text{sterms} [OF quality_increases_nsqn_fresh aodv_wf aodv_trans]}
\text{ oseq_{\text{step}}_invariant_\text{sterms} [OF oquality_increases \{i\} (orecvmsg msg_fresh), other quality_increases \{i\} \rightarrow]}
\text{ onl_{\text{invariant}}_\text{sterms} [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]}
\text{ onl_{\text{invariant}}_\text{sterms} [OF aodv_wf onl_\text{invariant_\text{sterms}} [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]]}
\text{ onl_{\text{invariant}}_\text{sterms} [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]}
\text{ simp del: One_nat_def, rule impl])

fix \(σ, \sigma'\) \(p l\)
assume "\((σ, p) \in \text{ooreachable} (opaodv i) ?S ?U\)"
and "\(1 \in \text{labels} \Gamma_{AODV} p\)"
and pre:
"\((l = \text{PAodv-:6} \lor l = \text{PAodv-:7} \lor l = \text{PRrep-:0} \lor l = \text{PRrep-:1}) \land 
-seql_onl_swap

\begin{proof}
\begin{quote}
(\text{cases "seql_onl_swap"})
assume "\((σ, i) \in \text{ooreachable} (opaodv i) ?S ?U\)"
and "\(1 \in \text{labels} \Gamma_{AODV} p\)" and hyp:
"\((l = \text{PAodv-:6} \lor l = \text{PAodv-:7} \lor l = \text{PRrep-:0} \lor l = \text{PRrep-:1}) \land 
-seql_onl_swap

\begin{proof}
\begin{quote}
(cases "seql_onl_swap")
assume "\((σ, i) \neq i\)"
from "\text{other quality_increases} \{i\} \ (σ, \sigma')" have "\text{quality_increases} (σ (\text{seql_onl_swap})) (σ' (\text{seql_onl_swap})))"
and hyp:
"\((l = \text{PAodv-:6} \lor l = \text{PAodv-:7} \lor l = \text{PRrep-:0} \lor l = \text{PRrep-:1}) \land 
-seql_onl_swap

\begin{proof}
\begin{quote}
next
assume "\((σ, i) = i\)" thus ?thesis
using \((σ', i) = σ i)\ hyp and pre by auto
qed
qed (\text{auto simp add: seqlsimp elim!: quality_increases_rreq_rrep Props})"
lemma rerr_sip:
"opaodv i |= (other with quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i} ->
onl Γ_AODV (λ(σ, l).
  l ∈ {PAodv:-8, PAodv:-9, PRerr:-0, PRerr:-1}
  → (∀ 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))"
(is "= (?S, ?U -> _)"
proof -
  { fix dests rip sip rsn and σ σ' :: "ip ⇒ state"
    assume qinc: "∀ j. quality_increases (σ j) (σ' j)"
    and *: "∀ ripc ∈ dom dests. ripc ∈ kD (rt (σ sip)) ∧
      the (dests ripc) - 1 ≤ nsqn (rt (σ sip)) ripc"
    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)) ripc"
    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)) ripc"
    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)) ripc" ..
      with ⟨rsn - 1 ≤ nsqn (rt (σ sip)) rip⟩ show "rsn - 1 ≤ nsqn (rt (σ’ sip)) ripc"
      by (rule le_trans)
    qed
  }
  note partial = this
  show ?thesis
    by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf
      oaodv_trans]
      onl_oainvariant_sterms [OF aodv_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: "paodv i |=
onl Γ_AODV (λ(ξ, l). (l = PRerr:-1
  → (∀ rip∈dom(dests ξ). rip ∈ kD(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 oaddpreRT_welldefined initiali_aodv oaodv_trans aodv_trans,
    simplified seql_onl_swap, 
    THEN oinvariant_anyact]

lemmas odests_vD_inc_sqn =
  open_seq_invariant [OF odests_vD_inc_sqn initiali_aodv oaodv_trans aodv_trans,
    simplified seql_onl_swap, 
    THEN oinvariant_anyact]

lemmas oprerr_guard =
  open_seq_invariant [OF oprerr_guard initiali_aodv oaodv_trans aodv_trans,
    simplified seql_onl_swap, 
    THEN oinvariant_anyact]

Proposition 7.28

lemma seq_compare_next_hop':

150
"opaodv i \models (otherwise quality_increases \{i\} (orecmsg msg_fresh),
other quality_increases \{i\} \rightarrow onl \Gamma_{AODV} (\lambda(\sigma, \_).
\forall dip. let nhip = the (nhop (rt (\sigma i)) dip)
in dip \in kD(rt (\sigma i)) \land nhip \neq dip \rightarrow
dip \in kD(rt (\sigma nhip)) \land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma nhip)) 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))) \land 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))) \land 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 \{dip\in kD(rt (\sigma (nhop dip)))\} have "dip\in kD (rt (\sigma' (nhop dip)))" ..
moreover have "nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"
proof -
from \{dip\in kD(rt (\sigma (nhop dip)))\}: qinc_nhop
have "nsqn (rt (\sigma (nhop dip))) dip \leq nsqn (rt (\sigma' (nhop dip))) dip" ..
with \{nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (nhop dip))) dip\}: show ?thesis
by simp
qed
ultimately show "dip\in kD(rt (\sigma' (nhop dip)))
\land 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)))
\land 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))))
\land 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)))
\land 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 \{dip\in kD(rt (\sigma i))\}: obtain dsn where "dests (\sigma i) dip = Some dsn"
by auto
with \{dip\in kD(rt (\sigma i))\}: 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 \{dests (\sigma i) dip = Some dsn\} have "the (dests (\sigma i) dip) = dsn" by simp
with ndest and \( \text{dip} \in \text{dom} \text{dests} (\sigma i) \), have "\( \text{dip} \in kD (\text{rt} (\sigma (\text{sip} (\sigma i))) ) \)"

\[ \text{dsn} - 1 \leq \text{nsqn} (\text{rt} (\sigma (\text{sip} (\sigma i)))) \]

by auto

moreover from issip and \( \text{dip} \in \text{dom} \text{dests} (\sigma i) \), have "\( \text{nhop} \text{dip} = \text{sip} (\sigma i) \) " ..

ultimately have "\( \text{dip} \in kD (\text{rt} (\sigma (\text{nhop} \text{dip}))) \)"

and "\( \text{dsn} - 1 \leq \text{nsqn} (\text{rt} (\sigma (\text{nhop} \text{dip}))) \)" by auto

with qinc show "\( \text{dsn} - 1 \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop} \text{dip}))) \)" by simp (metis kD_nsqn_quality_increases_trans)

qed

ultimately show ?thesis by simp

next

assume "\( \text{dip} \notin \text{dom} \text{dests} (\sigma i) \)"

with \( \text{dip} \in kD (\text{rt} (\sigma i)) \)

have "\( \text{nsqn} (\text{invalidate} (\text{rt} (\sigma i)) (\text{dests} (\sigma i))) \text{dip} = \text{nsqn} (\text{rt} (\sigma i)) \text{dip} \)"

by (rule nsqn_invalidate_other)

with \( \text{nsqn} (\text{rt} (\sigma i)) \text{dip} \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop} \text{dip}))) \) dip show ?thesis by simp

qed

with \( \text{dip} \in kD (\text{rt} (\sigma' (\text{nhop} \text{dip}))) \)

show "\( \text{dip} \in kD (\text{rt} (\sigma' (\text{nhop} \text{dip}))) \)

\wedge \text{nsqn} (\text{invalidate} (\text{rt} (\sigma i)) (\text{dests} (\sigma i))) \text{dip} \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop} \text{dip}))) \) dip" ..

qed

} note basic_prerr = this

{ fix \( \sigma \sigma' :: "\text{ip} \Rightarrow \text{state}" \)

assume a1 : "\( \forall \text{dip} \in kD (\text{rt} (\sigma i)) . \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip}) \neq \text{dip} \\
\rightarrow \text{dip} \in kD (\text{rt} (\sigma (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip})))) \\
\wedge \text{nsqn} (\text{rt} (\sigma i)) \text{dip} \leq \text{nsqn} (\text{rt} (\sigma (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip})))) \) dip"

and a2 : "\( \forall j . \text{quality\_increases} (\sigma j) (\sigma' j) \)"

have "\( \forall \text{dip} \in kD (\text{rt} (\sigma i)) . \\
\text{the} (\text{nhop} (\text{update} (\text{rt} (\sigma i)) (\text{sip} (\sigma i)) (0, \text{unk}, \text{val}, \text{Suc} 0, \text{sip} (\sigma i), \{\})) \text{dip}) \neq \text{dip} \\
\rightarrow \text{dip} \in kD (\text{rt} (\sigma' (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip})))) \\
\wedge \text{nsqn} (\text{update} (\text{rt} (\sigma i)) (\text{sip} (\sigma i)) (0, \text{unk}, \text{val}, \text{Suc} 0, \text{sip} (\sigma i), \{\})) \text{dip} \\
\leq \text{nsqn} (\text{rt} (\sigma' (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip})))) (\text{sip} (\sigma i)) \\
\rightarrow \text{dip} (\text{rt} (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip})))) \) dip"

by (drule(1) basic, auto)

thus "\( ?P \text{ dip} \) " by (cases "\( \text{dip} = \text{sip} (\sigma i) \)") auto

qed

} note nhop_update_sip = this

{ fix \( \sigma \sigma' \text{ oip sip osn hops} \)

assume pre : "\( \forall \text{dip} \in kD (\text{rt} (\sigma i)) . \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip}) \neq \text{dip} \\
\rightarrow \text{dip} \in kD (\text{rt} (\sigma (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip})))) \\
\wedge \text{nsqn} (\text{rt} (\sigma i)) \text{dip} \leq \text{nsqn} (\text{rt} (\sigma (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip})))) \) dip"

and qinc : "\( \forall j . \text{quality\_increases} (\sigma j) (\sigma' j) \)"

and * : "\( \text{sip} \neq \text{oip} \rightarrow \text{oip} \in kD (\text{rt} (\sigma \text{ sip})) \\
\wedge \text{osn} \leq \text{nsqn} (\text{rt} (\sigma \text{ sip})) \text{ oip} \\
\wedge (\text{nsqn} (\text{rt} (\sigma \text{ sip})) \text{ oip} = \text{osn} \\
\rightarrow \text{the} (\text{dhops} (\text{rt} (\sigma \text{ sip}))) \text{ oip} \leq \text{hops} \\
\vee \text{the} (\text{flag} (\text{rt} (\sigma \text{ sip}))) \text{ oip} = \text{inv}) \)"

from pre and qinc

have pre' : "\( \forall \text{dip} \in kD (\text{rt} (\sigma i)) . \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip}) \neq \text{dip} \\
\rightarrow \text{dip} \in kD (\text{rt} (\sigma' (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip})))) \\
\wedge \text{nsqn} (\text{rt} (\sigma i)) \text{dip} \leq \text{nsqn} (\text{rt} (\sigma' (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{dip})))) \) 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))))
    )" (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 (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) oip) ≠ dip
    → dip ∈ 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))))
    )" (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 (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) oip) ≠ 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, {})) oip) ≠ 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 \(\text{rtnot} \ qinc \ \text{dip} \in \text{kD}(rt \ (\sigma \ i)) \ \text{notdip} \ *
\)

have "?dip\_in\_kD \ dip"

by \(\text{simp (metis kD\_quality\_increases)}\)

moreover from \(\text{dip} = \text{oop} \ \text{rtnot} \ qinc \ \text{dip} \in \text{kD}(rt \ (\sigma \ i)) \ \text{notdip} \ *
\)

have "?nsqn\_le\_nsqn \ dip" by \(\text{simp (metis kD\_nsqn\_quality\_increases\_trans)}\)

ultimately show ?thesis ..

qed

qed

\}

note update2 = this

have "opaodv \ i \ = \ (\sigma, \ \text{U} \rightarrow) \ \text{onl} \ \Gamma_{AODV} \ (\lambda \sigma, \ _). \ 
\forall \text{dip} \in \text{kD}(rt \ (\sigma \ i), \ \text{the} \ (\text{nhop} \ (rt \ (\sigma \ i)) \ \text{dip}) \ \neq \ \text{dip} 
\rightarrow \ \text{dip} \in \text{kD}(rt \ (\sigma \ (\text{the} \ (\text{nhop} \ (rt \ (\sigma \ i)) \ \text{dip})))) 
\wedge \ \text{nsqn} \ (rt \ (\sigma \ i)) \ \text{dip} \ \leq \ \text{nsqn} \ (rt \ (\sigma \ (\text{the} \ (\text{nhop} \ (rt \ (\sigma \ i)) \ \text{dip})))) \ \text{dip})"

by \(\text{inv\_cterms inv add: oseq\_step\_invariant\_sterms \ [OF oquality\_increases\_nsqn\_fresh aodv\_wf oaedv\_trans]}\)

\text{onl\_oinvariant\_sterms \ [OF aodv\_wf oaddpreRT\_welldefined]}\)

\text{onl\_oinvariant\_sterms \ [OF aodv\_wf odests\_vD\_inc\_sqn]}\)

\text{onl\_oinvariant\_sterms \ [OF aodv\_wf oprerr\_guard]}\)

\text{onl\_oinvariant\_sterms \ [OF aodv\_wf rreq\_sip]}\)

\text{onl\_oinvariant\_sterms \ [OF aodv\_wf rrep\_sip]}\)

\text{onl\_oinvariant\_sterms \ [OF aodv\_wf rrerr\_sip]}\)

\text{other\_quality\_increases}\)

\text{other\_localD}\)

solve: \text{basic basic\_prerr}\)

\text{simpl add: sqnsinp nsqn\_invalidate nhop\_update\_sip}\)

\text{simpl del: One\_nat\_def}\)

\text{(rule conjI, erule(2) update1, erule(2) update2)+}\)

thus ?thesis unfolding \text{Let\_def} by auto

qed

Proposition 7.30

lemmas okD\_unk\_or\_atleast\_one =

\text{open\_seq\_invariant \ [OF kD\_unk\_or\_atleast\_one \ initiali\_aodv, simplified seql\_onl\_swap]}\)

lemmas ozero_seq\_unk\_hops\_one =

\text{open\_seq\_invariant \ [OF zero_seq\_unk\_hops\_one \ initiali\_aodv, simplified seql\_onl\_swap]}\)

lemma ooreachable\_fresh\_okD\_unk\_or\_atleast\_one:

fixes dip

assumes "\((\sigma, \ p) \in \ \text{oreachable} \ \text{(opaodv} \ i) \)

\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} \ \lor \ 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 \(\text{metis aodv\_ex\_label}\)

with \text{assms(1) have "\forall dip \in \text{kD}(rt \ (\sigma \ i)) \ ?P \ dip"}

by \(\text{(drule oinvariant\_weakenD \ [OF okD\_unk\_or\_atleast\_one \ [OF oaedv\_trans aodv\_trans]], auto dest!: otherwith\_actionD onlD simp: sqnsinp)}\)

with \(\text{dip} \in \text{kD}(rt \ (\sigma \ i))\) show ?thesis by \text{simp}\)

qed

lemma ooreachable\_fresh\_ozero_seq\_unk\_hops\_one:

fixes dip

assumes "\((\sigma, \ p) \in \ \text{oreachable} \ \text{(opaodv} \ i) \)

\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 "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 Γ AODV
p" by (metis aodv_ex_label)
with assms(1) have "∀dip ∈ kD (rt (σ i)). ?P dip"
  by -(drule invariant_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 Γ AODV (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 i)) dip = 1
∧ (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 i) dip = 1
∧ (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 i) dip = 1"
  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 simpsimp
  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 simpsimp
  with dip ∈ vD rt (σ i) dip = 1
  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 simpsimp
  with * show ?thesis by simp

qed

qed

} note basic = this
\{ \text{fix } \sigma \ a \ \text{dip} \ \text{sip} \ i \}
\begin{align*}
\text{assume } a1: & \ \forall \text{dip} \in \mathcal{V}(rt (\sigma \ i)) \\
& \hspace{1cm} \land \text{dip} \in \mathcal{V}(rt (\sigma \ (\text{the} (\text{nhop} (\sigma \ i))))) \\
& \hspace{1cm} \land \text{the} (\text{nhop} (rt (\sigma \ i)) \neq \text{dip}) \\
& \hspace{1cm} \rightarrow \text{rt} (\sigma \ i) \sqsubseteq_{\text{dip}} \text{rt} (\sigma \ (\text{the} (\text{nhop} (rt (\sigma \ i)))))
\end{align*}

\text{and ow: } "\text{if } \sigma \ a \ \text{dip} \ a"

\begin{align*}
\text{have } & \ \forall \text{dip} \in \mathcal{V}(\text{update} (rt (\sigma \ i)) \land \text{dip} \in \mathcal{V}(rt (\sigma \ i))) \\
& \hspace{1cm} \land \text{the} (\text{nhop} (\text{update} (rt (\sigma \ i)) \neq \text{dip}) \\
& \hspace{1cm} \rightarrow \text{update} (rt (\sigma \ i)) \sqsubseteq_{\text{dip}} \text{rt} (\sigma \ (\text{the} (\text{nhop} (rt (\sigma \ i)))))
\end{align*}

\text{proof clarify}
\text{fix dip}
\text{assume a2: } "\text{dip} \in \mathcal{V}(\text{update} (rt (\sigma \ i)) \land \text{dip} \in \mathcal{V}(rt (\sigma \ i)))" \\
\text{and a3: } "\text{dip} \in \mathcal{V}(rt (\sigma \ (\text{the} (\text{nhop} (rt (\sigma \ i)))))" \\
\text{and a4: } "\text{the} (\text{nhop} (\text{update} (rt (\sigma \ i)) \neq \text{dip}) \\
\text{show } "\text{update} (rt (\sigma \ i)) \sqsubseteq_{\text{dip}} \text{rt} (\sigma \ (\text{the} (\text{nhop} (rt (\sigma \ i)))))"

\text{proof (cases } "\text{dip} = \text{sip}"
\text{assume } "\text{dip} = \text{sip}"
\text{with } \text{the} (\text{nhop} (\text{update} (rt (\sigma \ i)) \neq \text{dip}) \neq \text{dip}) \rightarrow \text{False} \\
\text{thus } ?\text{thesis } ..

\text{next}
\text{assume [simp]: } "\text{dip} \neq \text{sip}"
\text{from a2 have } "\text{dip} \in \mathcal{V}(rt (\sigma \ i)) \lor \text{dip} = \text{sip}"
\text{by (rule } \mathcal{V}_{\text{update_val}})
\text{with } \text{dip} \neq \text{sip} \text{ have } "\text{dip} \in \mathcal{V}(rt (\sigma \ i))" \text{ by simp}
\text{moreover from a3 have } "\text{dip} \in \mathcal{V}(rt (\sigma \ (\text{the} (\text{nhop} (rt (\sigma \ i)))))" \text{ by simp}
\text{moreover from a4 have } "\text{the} (\text{nhop} (rt (\sigma \ i)) \neq \text{dip})" \text{ by simp}
\text{ultimately have } "\text{rt} (\sigma \ i) \sqsubseteq_{\text{dip}} \text{rt} (\sigma \ (\text{the} (\text{nhop} (rt (\sigma \ i)))))"
\text{using a1 ow by } \text{- (drule(1) basic, simp)}
\text{with } \text{dip} \neq \text{sip} \text{ show } ?\text{thesis }
\text{by } \text{- (erule rt_strictly_fresher_update_other, simp)}
\text{qed}
\text{qed}
\}\text{ note update_0_unk = this}

\{ \text{fix } \sigma \ a \ \text{dip} \ \text{nhop} \}
\text{assume pre: } "\forall \text{dip} \in \mathcal{V}(rt (\sigma \ i)) \land \text{dip} \in \mathcal{V}(rt (\sigma \ (\text{nhop} \text{ dip}))) \land \text{nhop} \text{ dip} \neq \text{dip} \\
\rightarrow \text{rt} (\sigma \ i) \sqsubseteq_{\text{dip}} \text{rt} (\sigma \ (\text{nhop} \text{ dip}))"
\text{and ow: } "\text{if } \sigma \ a \ \text{dip} \ a"
\text{have } "\forall \text{dip} \in \mathcal{V}(\text{invalidate} (rt (\sigma \ i)) \text{ dests} (\sigma \ i)) \\
\land \text{dip} \in \mathcal{V}(rt (\sigma \ (\text{nhop} \text{ dip}))) \land \text{nhop} \text{ dip} \neq \text{dip} \\
\rightarrow \text{rt} (\sigma \ i) \sqsubseteq_{\text{dip}} \text{rt} (\sigma \ (\text{nhop} \text{ dip}))"

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

\{ \text{fix } \sigma \ a \ \text{dip} \ \text{dip} \ \text{osn} \ \text{sip} \ \text{hops} \ i \}
assume pre: 
"\forall dip. \ dip \in \nuD (rt (\sigma \ i))
\land \ dip \in \nuD (rt (\sigma (the (nhop (rt (\sigma \ i)) dip))))
\land the (nhop (rt (\sigma \ i)) dip) \neq dip
\rightarrow rt (\sigma \ i) \sqcap dip rt (\sigma (the (nhop (rt (\sigma \ i)) dip)))"

and ow: "\exists i \in \sigma \ a"
and "Suc 0 \leq osn"
and a6: "sip \neq oip \rightarrow oip \in k\nuD (rt (\sigma \ sip))
\land osn \leq nsqn (rt (\sigma \ sip)) oip
\land (nsqn (rt (\sigma \ sip)) oip = osn
\rightarrow the (dhops (rt (\sigma \ sip)) oip) \leq hops
\lor 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 \nuD (update (rt (\sigma \ i)) oip (osn, kno, val, Suc hops, sip, \{\}))
\land dip \in \nuD (rt (\sigma ' (the (nhop (update (rt (\sigma \ i)) oip
(osn, kno, val, Suc hops, sip, \{\})) dip))))

\land 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, \{\})
\sqcap dip
rt (\sigma ' (the (nhop (update (rt (\sigma \ i)) oip (osn, kno, val, Suc hops, sip, \{\})) dip))))"

proof clarify
fix dip
assume a2: "\exists dip \in \nuD (update (rt (\sigma \ i)) oip (osn, kno, val, Suc (hops), sip, \{\}))"

and a3: "\exists dip \in \nuD (rt (\sigma ' (the (nhop (update (rt (\sigma \ i)) oip
(osn, kno, val, Suc hops, sip, \{\})) dip))))"
and a4: "\exists dip (the (nhop (update (rt (\sigma \ i)) oip (osn, kno, val, Suc hops, sip, \{\})) dip) \neq dip"

from ow have a5: "\exists j. j \neq i \rightarrow \sigma \ j = \sigma ' \ j " by auto
show "update (rt (\sigma \ i)) oip (osn, kno, val, Suc hops, sip, \{\})"
\sqcap dip
rt (\sigma ' (the (nhop (update (rt (\sigma \ i)) oip (osn, kno, val, Suc hops, sip, \{\})) dip)))"
(is "\exists dip \sqcap dip rt (\sigma ' (the (nhop (update (rt (\sigma \ i)) dip))))"

proof (cases "\exists dip = 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 "\exists dip \in \nuD (rt (\sigma \ i))" by simp
moreover from a3 have "\exists dip \in \nuD (rt (\sigma (the (nhop (rt (\sigma \ i)) dip))))"
using nochange and "\exists dip \sqcap dip rt (\sigma ' (the (nhop (rt (\sigma \ i)) dip)))"
moreover from a4 have "\exists dip (the (nhop (rt (\sigma \ i)) dip) \neq dip " by simp
ultimately have "rt (\sigma \ i) \sqcap dip rt (\sigma (the (nhop (rt (\sigma \ i)) dip)))"
using pre by simp

hence "rt (\sigma \ i) \sqcap dip rt (\sigma ' (the (nhop (rt (\sigma \ i)) dip)))"
using (\forall j. \sigma \ j = \sigma ' \ j) by simp
thus "\exists dip rt (\sigma ' (the (nhop (rt (\sigma \ i)) dip)))"

next
assume change: "\exists dip \neq rt (\sigma \ i)"
from after a2 have "\exists dip \in k\nuD (rt (\sigma ' \ i))" by auto
show ?thesis
proof (cases "dip = oip"
assume "dip \neq oip"

with a2 have "\exists dip \in \nuD (rt (\sigma \ i))" by auto
moreover with a3 a5 after and (dip \neq oip)
have "\exists dip \in \nuD (rt (\sigma (the (nhop (rt (\sigma \ i)) dip)))))"
by simp metis
moreover from a4 and (dip \neq oip) have "the (nhop (rt (\sigma \ i)) dip) \neq dip " by simp
ultimately have "rt (\sigma \ i) \sqcap dip rt (\sigma (the (nhop (rt (\sigma \ i)) dip)))"
using pre by simp

with after and a5 and (dip \neq oip) show ?thesis
by simp (metis rt_strictly_fresher_update_other
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(I) 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 \( oip \in kD(rt (\sigma i)) \)
    show "\((\lambda a. if a = oip then Some (\tau i) oip) else \tau i a) = \tau i)\)"
    by (auto dest!: kD_Some)
qed with change have False ..
thus ?thesis ..
qed

next assume "\( oip \in \iD(\tau i)\)"
with \(\langle\text{the (flag (\tau i) oip) = val}\rangle\) and a5 have "\( \text{ SIP = i}\)"
by (metis f.distinct(1) \(\iD\text{_flag_is_inv}\))
from \( \iop \in \iD(\tau i) \) have "\(\text{the (flag (\tau i) oip) = inv}\)" by auto
with \(\text{ SIP = i}\) \(\text{Suc 0 < osn}\) change after next
have "\(\pi oip < \pi (\tau i) oip\)"
unfolding update_def
by (clarsimp split: option.split_asm split_if_asm)
(auto simp: sqn_def)
with \(\text{ SIP = i}\) change after
have "\(\pi oip < \pi (\tau i) oip\)"
by simp thus ?thesis ..
qed

ultimately have "?rt1 ⊑ oip ?rt2 oip"
by (rule rt_strictly_fresher_eqI)
with (dip = oip) show ?thesis by simp
qed

have "opaodv i \models \{\text{otherwith } (op=) \} \{\text{orecvmsg } (\lambda \sigma m. \text{msg_fresh } \sigma m \land \text{msg_zhops } m)\},
other quality_increases \{i\} \to"
\begin{align*}
onl \Gamma_{\text{AODV}}
(\lambda (\sigma, \_). \forall \text{dip}. \text{dip} \in vD (rt \ (\sigma \ i)) \cap vD (rt \ (\sigma \ (\text{the (nhop (rt \ (\sigma \ i)) dip))})
\land \ (\text{the (nhop (rt \ (\sigma \ i)) dip)}) \neq \text{dip}
\to rt \ (\sigma \ i) \sqsubseteq dip rt \ (\sigma \ (\text{the (nhop (rt \ (\sigma \ i)) dip))}))"
\end{align*}
proof (inv_cterms inv add: onl_oinvariant_sterms [OF aodv_wf rreq_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf rrep_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf rrerr_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf oosn_rreq [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf odsn_rrep [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf oaddpreRT_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 (opaodv i) (?S i) (other quality_increases \{i\})}"
and "other quality_increases \{i\} \sigma'"
and ll: "l \in \text{labels } \Gamma_{\text{AODV}} \ p"
and pre: "\forall \text{dip}. \text{dip} \in vD (rt \ (\sigma \ i))
\land \ (\text{the (nhop (rt \ (\sigma \ i)) dip)}) \neq \text{dip}
\to rt \ (\sigma \ i) \sqsubseteq dip rt \ (\sigma \ (\text{the (nhop (rt \ (\sigma \ i)) dip))})"

from this(1-2)
have or': "(\sigma', p) \in \text{oreachable (opaodv i) (?S i) (other quality_increases \{i\})}"
by - (rule ooreachable_other')

from or and ll have next_hop: "\forall \text{dip}. \text{let nhip = the (nhop (rt \ (\sigma \ i)) dip)}
in dip \in kD(rt \ (\sigma \ i)) \land nhip \neq dip
\to dip \in kD(rt \ (\sigma \ nhip))
\land \text{nsqn (rt \ (\sigma \ i)) dip} \leq \text{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 \text{dip} \in kD (rt \ (\sigma \ i)) \cdot \text{sqn (rt \ (\sigma \ i)) dip} = 0
\to sqnf (rt \ (\sigma \ i)) dip = \text{unk}
\land \text{(dhops (rt \ (\sigma \ i)) dip)} = 1
\land \text{(the (dhops (rt \ (\sigma \ i)) dip))} = \text{dip}"
by (auto dest!: onl_oinvariant_weakenD [OF ozero_seq_unk_hops_one
[OF oaodv_trans aodv_trans])

otherwith_actionD
simp: seqlsimp)

from 'other quality_increases \{i\} \sigma \ \sigma' have "\sigma' \ i = \sigma \ i" by auto
hence "\text{quality_increases } (\sigma \ i) \ (\sigma' \ i)" by auto
with 'other quality_increases \{i\} \sigma \ \sigma' have "\forall \text{j. quality_increases } (\sigma \ j) \ (\sigma' \ j)"
by - (erule otherE, metis singleton_iff)

show "\forall \text{dip}. \text{dip} \in vD (rt \ (\sigma' \ i))
\land \text{dip} \in vD (rt \ (\sigma' \ (\text{the (nhop (rt \ (\sigma' \ i)) dip))})
\land \ (\text{the (nhop (rt \ (\sigma' \ i)) dip)}) \neq \text{dip}
\to rt \ (\sigma' \ i) \sqsubseteq dip rt \ (\sigma' \ (\text{the (nhop (rt \ (\sigma' \ i)) dip}))"
proof clarify
fix dip
assume "dip∈D(rt (σ' i))"
and "dip∈D(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∈D(rt (σ i))"
and "dip∈D(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∈D(rt (σ i))⟩ and next_hop
have "dip∈D(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∈D(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∈D(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∈D(rt (σ ?nhip)))")
assume "dip∈D(rt (σ ?nhip))"
with ⟨dip∈D(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∈D(rt (σ ?nhip))⟩
by (rule strictly_fresher_quality_increases_right)
next
assume "dip∈D(rt (σ ?nhip))"
with ⟨dip∈D(rt (σ ?nhip))⟩ have "dip∈D(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∈D(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∈D(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and (σ' i = σ i)
show "dip∈D(rt (σ' ?nhip))" by simp
qed
finally have "nsqn (rt (σ i)) dip < nsqn (rt (σ' ?nhip)) dip" .
moreover from ⟨dip∈D(rt (σ' (the (nhop (rt (σ' i)) dip))))⟩ and (σ' i = σ i)
have "$\text{dip} \in kD(\text{rt} (\sigma' \ ?\text{nhip}))" by auto
ultimately show "$\text{rt} (\sigma \ i) \sqsubseteq \text{dip} \ \text{rt} (\sigma' \ ?\text{nhip})"
using (\text{dip} \in kD(\text{rt} (\sigma \ i))) by - (\text{rule} \ \text{rt}\text{-strictly}\text{-fresher}\_\text{ltI})
qed

with (\sigma' \ i = \sigma \ i) show "$\text{rt} (\sigma' \ i) \sqsubseteq \text{dip} \ \text{rt} (\sigma' \ \text{the} (\text{nhop} (\text{rt} (\sigma' \ i)) \ \text{dip}))"
by simp
qed

qed

thus ?thesis unfolding \text{Let}\_\text{def}.

qed

lemma \text{seq}\_\text{compare}\_\text{next}\_\text{hop}:
fixes \ w
shows "\text{opaodv} \ i \models (\text{otherwith} (\text{op}=) \ \{i\} \ (\text{orecvmsg} \ \text{msg}\_\text{fresh}),
\text{other}\_\text{quality}\_\text{increases} \ \{i\} \rightarrow)
\text{global} (\lambda \sigma. \ \forall \text{dip}. \ \text{let} \ \text{nhip} = \text{the} (\text{nhop} (\text{rt} (\sigma \ i)) \ \text{dip})
in \text{dip} \in kD(\text{rt} (\sigma \ i)) \ \land \ \text{nhip} \neq \text{dip} \rightarrow
\text{dip} \in kD(\text{rt} (\sigma \ \text{nhip}))
\land \ \text{nsqn} (\text{rt} (\sigma \ i)) \ \text{dip} \leq \ \text{nsqn} (\text{rt} (\sigma \ \text{nhip})) \ \text{dip})"
by (\text{rule} \ \text{o}\text{invariant}\_\text{weakenE} [OF \ \text{seq}\_\text{compare}\_\text{next}\_\text{hop}']) (\text{auto} \ \text{dest!}: \ \text{onlD})

lemma \text{seq}\_\text{nhop}\_\text{quality}\_\text{increases}:
shows "\text{opaodv} \ i \models (\text{otherwith} (\text{op}=) \ \{i\} \ (\text{orecvmsg} \ (\lambda \sigma \ m. \ \text{msg}\_\text{fresh} \ \sigma \ m \ \land \ \text{msg}\_\text{zhops} \ m)),
\text{other}\_\text{quality}\_\text{increases} \ \{i\} \rightarrow)
\text{global} (\lambda \sigma. \ \forall \text{dip}. \ \text{let} \ \text{nhip} = \text{the} (\text{nhop} (\text{rt} (\sigma \ i)) \ \text{dip})
in \text{dip} \in vD (\text{rt} (\sigma \ i)) \ \land \ vD (\text{rt} (\sigma \ \text{nhip})) \ \land \ \text{nhip} \neq \text{dip}
\rightarrow (\text{rt} (\sigma \ i)) \ \sqsubseteq \text{dip} (\text{rt} (\sigma \ \text{nhip})))"
by (\text{rule} \ \text{o}\text{invariant}\_\text{weakenE} [OF \ \text{seq}\_\text{nhop}\_\text{quality}\_\text{increases}']) (\text{auto} \ \text{dest!}: \ \text{onlD})

end

1.10 Routing graphs and loop freedom

theory \ A\_\text{Loop}\_\text{Freedom}
imports \ A\_\text{Aodv}\_\text{Predicates} \ A\_\text{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 \text{state}) \Rightarrow \text{ip} \Rightarrow \text{ip} \ \text{rel}"
where
"\rt\_graph \ \sigma = (\lambda \text{dip}.
\{(\text{ip}, \ \text{ip}') \mid \text{ip} \ \text{ip'} \ \text{dsn} \ \text{dsk} \ \text{hops} \ \text{pre}.
\text{ip} \neq \text{dip} \ \land \ \text{rt} (\sigma \ \text{ip}) \ \text{dip} = \text{Some} (\text{dsn}, \ \text{dsk}, \ \text{val}, \ \text{hops}, \ \text{ip'}, \ \text{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\_graph\_E [elim]:
fixes \ n \ dip \ ip \ ip'
assumes "(\text{ip}, \ \text{ip'}) \in \rt\_graph \ \sigma \ \text{dip}"
shows "\text{ip} \neq \text{dip} \ \land \ (\exists \ \text{rt} (\sigma \ \text{ip}) = \text{r}
\land \ (\exists \ \text{dsn} \ \text{dsk} \ \text{hops} \ \text{pre}. \ \text{r} \ \text{dip} = \text{Some} (\text{dsn}, \ \text{dsk}, \ \text{val}, \ \text{hops}, \ \text{ip'}, \ \text{pre})))"
using assms unfolding \rt\_graph\_def by auto

lemma \rt\_graph\_vD [dest]:
"\forall \text{ip}' \sigma \ \text{dip}. (\text{ip}, \ \text{ip}') \in \rt\_graph \ \sigma \ \text{dip} \ \Longrightarrow \ \text{dip} \in vD(\text{rt} (\sigma \ \text{ip}))"
unfolding \rt\_graph\_def \ vD\_def by auto

lemma \rt\_graph\_vD\_trans [dest]:
"\forall \text{ip}' \sigma \ \text{dip}. (\text{ip}, \ \text{ip}') \in (\rt\_graph \ \sigma \ \text{dip})^+ \ \Longrightarrow \ \text{dip} \in vD(\text{rt} (\sigma \ \text{ip}))"
by (erule converse\_tranclE) auto

162
lemma rt_graph_not_dip [dest]:
"\forall \text{ip} \ \text{ip'} \ \text{σ} \ \text{dip}. (\text{ip}, \text{ip'}) \in \text{rt}\_\text{graph} \ \text{σ} \ \text{dip} \implies \text{ip} \neq \text{dip}"
unfolding rt_graph_def by auto

lemma rt_graph_not_dip_trans [dest]:
"\forall \text{ip} \ \text{ip'} \ \text{σ} \ \text{dip}. (\text{ip}, \text{ip'}) \in (\text{rt}\_\text{graph} \ \text{σ} \ \text{dip})^+ \implies \text{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]:
"\forall \text{ip} \ \text{ip'} \ \text{σ} \ \text{dip}. (\text{ip}, \text{ip'}) \in \text{rt}\_\text{graph} \ \text{σ} \ \text{dip} \implies \text{ip'} = \text{the (nhop (rt (σ ip)) dip)}"
unfolding rt_graph_def by auto

theorem inv_to_loop_freedom:
assumes "\forall i \ \text{dip}. \text{let nhip} = \text{the (nhop (rt (σ i)) dip) in dip ∈ \text{vD (rt (σ i))} \cap \text{vD (rt (σ nhip))} ∧ \text{nhip} \neq \text{dip} \rightarrow (rt (σ i) \sqsubseteq \text{dip} (rt (σ nhip)))"
shows "\forall \text{dip}. \text{irrefl ((rt}\_\text{graph} \ \text{σ} \ \text{dip})^+)"
using assms proof (intro allI)
  fix σ :: "\text{ip} \Rightarrow \text{state}" and dip
  assume inv: "\forall \text{ip} \ \text{dip}. \text{let nhip} = \text{the (nhop (rt (σ ip)) dip) in dip ∈ \text{vD (rt (σ ip))} \cap \text{vD (rt (σ nhip))} ∧ \text{nhip} \neq \text{dip} \rightarrow (rt (σ ip) \sqsubseteq \text{dip} (rt (σ nhip)))"

  { fix ip ip' assume "\text{(ip, ip')} ∈ (\text{rt}\_\text{graph} \ \text{σ} \ \text{dip})^+
    and \text{dip} ∈ \text{vD (rt (σ ip'))}
    and \text{ip'} \neq \text{dip}
    hence "\text{rt (σ ip)} \sqsubseteq \text{dip rt (σ ip')}"
    proof induction
      fix nhip
      assume "\text{(ip, nhip)} ∈ \text{rt}\_\text{graph} \ \text{σ} \ \text{dip}
      and \text{dip} ∈ \text{vD (rt (σ nhip))}
      and \text{nhip} \neq \text{dip}
      from ⟨\text{(ip, nhip)} ∈ \text{rt}\_\text{graph} \ \text{σ} \ \text{dip}⟩ have \text{dip} ∈ \text{vD (rt (σ ip))} and \text{nhip = the (nhop (rt (σ ip)) dip)}
      and \text{dip} ∈ \text{vD (rt (σ nhip))}
      and inv
      show "\text{rt (σ ip)} \sqsubseteq \text{dip rt (σ nhip)}"
      by (clarsimp simp: Let_def)
      next
      fix nhip nhip'
      assume "\text{(ip, nhip)} ∈ (\text{rt}\_\text{graph} \ \text{σ} \ \text{dip})^+
      and \text{nhip, nhip'} ∈ \text{rt}\_\text{graph} \ \text{σ} \ \text{dip}
      and IH: "\text{[dip ∈ \text{vD (rt (σ nhip))}; \text{nhip} \neq \text{dip}] \implies rt (σ ip) \sqsubseteq \text{dip rt (σ nhip)}"
      and "\text{dip} ∈ \text{vD (rt (σ nhip'))}
      and "\text{nhip'} \neq \text{dip}
      from ⟨\text{nhip, nhip'} ∈ \text{rt}\_\text{graph} \ \text{σ} \ \text{dip}⟩ have 1: "\text{dip} ∈ \text{vD (rt (σ nhip))} and "\text{nhip} = \text{the (nhop (rt (σ nhip)) dip)}
      2: "\text{nhip} \neq \text{dip}
      and "\text{nhip'} = \text{the (nhop (rt (σ nhip)) dip)}
      by auto
      from 1 2 have "\text{rt (σ ip)} \sqsubseteq \text{dip rt (σ nhip)}" by (rule IH)
      also have "\text{rt (σ nhip)} \sqsubseteq \text{dip rt (σ nhip')}"
      proof -
      from \text{dip} ∈ \text{vD (rt (σ nhip))} and \text{dip} ∈ \text{vD (rt (σ nhip'))}
      have "\text{dip} ∈ \text{vD (rt (σ nhip))} \cap \text{vD (rt (σ nhip'))}"
      and "\text{nhip} \neq \text{dip}
      and "\text{nhip'} = \text{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 ip)".
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

1.11 Lift and transfer invariants to show loop freedom

theory A_Aodv_Loop_Freedom
imports "../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 Γ AODV i) (seqp_sos Γ QMSG)"
  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 | ⟨⟨ 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 (⟨⟨ i qmsg | (orecvmsg (λ σ 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 "call Γ AODV (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!: ostepD)
  qed
  qed

next
fix σ σ' a
assume "otherwith (op=) {i} (orecvmsg (λ σ 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

lemma par_rreq_rrep_sn_quality_increases:
  "opaodv i (⟨⟨ i qmsg | ⟨⟨ i qmsg, other (λ_. True) {i} →)
    globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))")
  proof -
have "opaodv i \=_{A} (\lambda_\sigma \cdot \text{orevcmsg}\ (\lambda_. \text{req\_rrep\_sn})\ \sigma), \text{other}\ (\lambda_. \cdot \text{True})\ \{i\} \rightarrow \text{globala}\ (\lambda(\sigma, a, \sigma').\ \text{quality\_increases}\ \{\sigma\ i\ (\sigma'\ i)\})"
by (rule ostep\_invariant\_weakenE \[OF olocal\_quality\_increases\])
(auto dest!: aodv\_ex\_label)

hence "opaodv i \=_{A} (\lambda_\sigma \cdot \text{orevcmsg}\ (\lambda_. \text{req\_rrep\_sn})\ \sigma), \text{other}\ (\lambda_. \cdot \text{True})\ \{i\} \rightarrow \text{globala}\ (\lambda(\sigma, a, \sigma').\ \text{quality\_increases}\ \{\sigma\ i\ (\sigma'\ i)\})"
by (rule lift\_step\_into\_qmsg\_stateless\_assm) simp\_all
thus \?thesis by rule auto

qed

lemma \text{par\_req\_rrep\_nsqn\_fresh\_any\_step}: shows "opaodv i \=_{A} (\lambda_\sigma \cdot \text{orevcmsg}\ (\lambda_. \text{req\_rrep\_sn})\ \sigma), \text{other}\ (\lambda_. \cdot \text{True})\ \{i\} \rightarrow \text{globala}\ (\lambda(\sigma, a, \sigma').\ \text{anycast}\ (\text{msg\_fresh}\ \sigma)\ a)"
proof -
have "opaodv i \=_{A} (\lambda_\sigma \cdot \text{orecvmsg}\ (\lambda_. \text{req\_rrep\_sn}))\ \sigma), \text{other}\ (\lambda_. \cdot \text{True})\ \{i\} \rightarrow \text{globala}\ (\lambda(\sigma, a, \sigma').\ \text{anycast}\ (\text{msg\_fresh}\ \sigma)\ a)"
proof (rule ostep\_invariant\_weakenE \[OF qreq\_rrep\_nsqn\_fresh\_any\_step\_invariant\])
fix t
assume "\text{onll}\ \Gamma_{AODV}\ \{\lambda(\sigma, a, \cdot)\ . \text{anycast}\ (\text{msg\_fresh}\ \sigma)\ a\ t\}"
thus "\text{globala}\ \{\lambda(\sigma, a, \sigma')\ . \text{anycast}\ (\text{msg\_fresh}\ \sigma)\ a\ t\}"

by (cases t) (clarsimp dest!: aodv\_ex\_label)

qed auto

hence "opaodv i \=_{A} (\lambda_\sigma \cdot \text{orecvmsg}\ (\lambda_. \text{req\_rrep\_sn}))\ \sigma), \text{other}\ (\lambda_. \cdot \text{True})\ \{i\} \rightarrow \text{globala}\ (\lambda(\sigma, a, \sigma').\ \text{anycast}\ (\text{msg\_fresh}\ \sigma)\ a)"
by (rule lift\_step\_into\_qmsg\_stateless\_assm) simp\_all
thus \?thesis by rule auto

qed

lemma \text{par\_anycast\_msg\_zhops}: shows "opaodv i \=_{A} (\lambda_\sigma \cdot \text{orevcmsg}\ (\lambda_. \text{req\_rrep\_sn})\ \sigma), \text{other}\ (\lambda_. \cdot \text{True})\ \{i\} \rightarrow \text{globala}\ \{\lambda(\cdot, a, \cdot)\ . \text{anycast}\ \text{msg\_zhops}\ a\}"
proof -
from \text{anycast\_msg\_zhops}\ \text{initial\_aodv}\ \text{oaodv\_trans}\ \text{aodv\_trans}
have "opaodv i \=_{A} (\lambda_\sigma \cdot \text{act}\ \text{TT}, \text{other}\ (\lambda_. \cdot \text{True})\ \{i\} \rightarrow \text{seqll} i (\text{onll}\ \Gamma_{AODV}\ \{\lambda(\cdot, a, \cdot)\ . \text{anycast}\ \text{msg\_zhops}\ a\})"
by (rule open\_seq\_step\_invariant)

hence "opaodv i \=_{A} (\lambda_\sigma \cdot \text{orevcmsg}\ (\lambda_. \text{req\_rrep\_sn})\ \sigma), \text{other}\ (\lambda_. \cdot \text{True})\ \{i\} \rightarrow \text{globala}\ \{\lambda(\cdot, a, \cdot)\ . \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}\ \text{label})\ \text{seqp}), \text{msg}\ \text{seq\_action})\ \text{transition}"
assume "\text{seqll} i (\text{onll}\ \Gamma_{AODV}\ \{\lambda(\cdot, a, \cdot)\ . \text{anycast}\ \text{msg\_zhops}\ a\})\ t"
thus "\text{globala}\ \{\lambda(\cdot, a, \cdot)\ . \text{anycast}\ \text{msg\_zhops}\ a\}\ t"
by (cases t) (clarsimp dest!: seqll\_D onll\_D, metis aodv\_ex\_label)

qed simp\_all

hence "opaodv i \=_{A} (\lambda_\sigma \cdot \text{orevcmsg}\ (\lambda_. \text{req\_rrep\_sn})\ \sigma), \text{other}\ (\lambda_. \cdot \text{True})\ \{i\} \rightarrow \text{globala}\ \{\lambda(\cdot, a, \cdot)\ . \text{anycast}\ \text{msg\_zhops}\ a\}"
by (rule lift\_step\_into\_qmsg\_stateless\_assm) simp\_all
thus \?thesis by rule auto

qed

11.1.1 Lift to nodes

lemma \text{node\_step\_no\_change\_on\_send\_or\_receive}: assumes "((\sigma, \text{NodeS\ i\ P\ R}), a, (\sigma', \text{NodeS\ i'\ P'\ R'})) \in\ \text{node\_sos}\ (\text{oparp\_sos}\ i\ \text{osqep\_sos}\ \Gamma_{AODV}\ i\ \text{seqp\_sos}\ \Gamma_{QMSG})"
and "a \neq \tau"
shows "\sigma' = \sigma\ i"
using \text{assms}
by (cases a) (auto elim!: par\_step\_no\_change\_on\_send\_or\_receive)

lemma \text{node\_nhop\_quality\_increases}: shows "\{ i : opaodv i \=_{A} \{i\} \rightarrow \{ \text{otherwith}\ (\text{op=}\ \{i\} \}

\text{node\_nhop\_quality\_increases} :
(oarrivemsg (\lambda \sigma. \msg_fresh \sigma \land \msg_zhops \sigma)),
   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)) \land nhip \neq \dip
\rightarrow (rt (\sigma \ i)) \subseteq dip (rt (\sigma \ nhip)))"}

by (rule node_lift [OF \par_nhop_quality_increases]) auto

lemma \node_quality_increases:
"(i : \opaadv i \in \{i \ \qmsg : R \}_o) \models (\lambda \sigma. \oarrivemsg (\lambda \_ . \ \reqrep \_sn) \sigma,
   other (\lambda \_ . True) \{i\} \rightarrow)
globala (\lambda (\sigma, a, \sigma'). \quality_increases (\sigma \ i) (\sigma' \ i))"}

by (rule node_lift_step_statelessasm [OF \par_reqrep_sn_quality_increases]) simp

lemma \node_rreq_rrep_nsqn_fresh_any_step:
shows "(i : \opaadv i \in \{i \ \qmsg : R_i\}_o) \models (\lambda \sigma. \oarrivemsg (\lambda \_ . \ \reqrep \_sn) \sigma,
   other (\lambda \_ . True) \{i\} \rightarrow)
globala (\lambda (\sigma, a, \_). \castmsg (\msg_fresh \sigma) a)"

by (rule node_lift_anycast_statelessasm [OF \par_rreq_rrep_nsqn_fresh_any_step])

lemma \node_silent_change_only:
shows "(i : \opaadv i \in \{i \ \qmsg : R_i\}_o) \models (\lambda \sigma. \oarrivemsg (\lambda \_ . True) \sigma,
   other (\lambda \_ . True) \{i\} \rightarrow)
globala (\lambda (\_, a, \_). a \neq \tau \rightarrow \sigma' \ i = \sigma \ i)"

proof (rule ostep_invariantI, simp (no_asm), rule impI)

fix \sigma \ \zeta \ a \ \sigma' \ \sigma''

assume or: "(\sigma, \zeta) \in \oreachable ((i : \opaadv i \in \{i \ \qmsg : R_i\}_o) (\lambda \sigma. \oarrivemsg (\lambda \_ . True) \sigma,
   other (\lambda \_ . True) \{i\} \rightarrow)
globala (\lambda (\sigma, a, \_). a \neq \tau \rightarrow \sigma' \ i = \sigma \ i)"

and tr: "((\sigma, \zeta), a, (\sigma', \zeta')) \in \trans ((i : \opaadv i \in \{i \ \qmsg : R_i\}_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, \NodeS i p R), a, (\sigma', \zeta')) \in \onode_sos (\opaarp_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

1.11.2 Lift to partial networks

lemma \arrive_rreq_rrep_nsqn_fresh_inc_sn [simp]:
assumes "oarrivemsg (\lambda \sigma. \msg_fresh \sigma \land \msg_m)\ \sigma \ m"

shows "oarrivemsg (\lambda \_. \ \reqrep \_sn) \sigma \ m"

using assms by (cases m) auto

lemma \opnet_nhop_quality_increases:
shows "\opnet (\lambda i. \opaadv i \in \{i \ \qmsg\} \ p) =
   (\otherwith (\op =) (\net_tree_ips \ p))
   (oarrivemsg (\lambda \sigma. \msg_fresh \sigma \land \msg_zhops \sigma)),
   other \quality_increases (\net_tree_ips \ p) \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)) \land nhip \neq \dip
\rightarrow (rt (\sigma \ i)) \subseteq dip (rt (\sigma \ nhip)))"

proof (rule pnet_lift [OF \par_nhop_quality_increases])

fix \ i \ R

166
have "(i : oapadv i (⟨i⟩ qmsg : R)ᵦ) ⇒ (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 a σ' s'
assume or: "(σ, σ) ∈ oreachable ((i : oapadv i (⟨i⟩ qmsg : R)ᵦ)
(λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ)
other (λ_. True) (i))"
and tr: "((σ, σ), a, (σ', s')) ∈ trans ((i : oapadv i (⟨i⟩ qmsg : R)ᵦ)"
and am: "oarrivemsg (λ_. rreq_rrep_sn) σ a"

proof
fix i R
"(i : oapadv i (⟨i⟩ qmsg : R)ᵦ) ⇒
(λσ _. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
other quality_increases {i} →) globala (λ(σ, a, σ').)
castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)"
by rule auto

qed

thus "(i : oapadv i (⟨i⟩ qmsg : R)ᵦ) ⇒
(λσ _. oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
other quality_increases {i} →) globala (λ(σ, a, σ').)
castmsg (λm. msg_fresh σ m ∧ msg_zhops m) a)" by (case_tac a) auto

next
fix i R
show "(i : oapadv i (⟨i⟩ qmsg : R)ᵦ) ⇒
(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_rreq_rrep_msgn_fresh_any_step]) auto

next
fix i R
show "(i : oapadv i (⟨i⟩ qmsg : R)ᵦ) ⇒
(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

1.1.3 Lift to closed networks

lemma onet_nhopping_quality_increases:
shows "oclosed (opnet (λi. oapadv 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_nhopping_quality_increases
shows "opnet (λi. oapadv 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 otherwhithE)
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 $\langle$inclosed $\sigma$ a$\rangle$ obtain d di where "ms = newpkt(d, di)"

by (cases ms) auto

ultimately show ?thesis by simp

qed simp_all

qed

1.11.4 Transfer into the standard model

interpretation aodv_openproc: openproc paodv opaodv id

where "aodv_openproc.initmissing = initmissing"

proof -

show "openproc paodv opaodv id"

proof unfold_locales

fix i :: ip

have "\{ (\sigma, \zeta). (\sigma i, \zeta) \in \sigma_{AODV} \land (\forall j. j \neq i \rightarrow \sigma j \in \text{fst} \ ' \ \sigma_{AODV} j) \} \subseteq \sigma_{AODV} '"

unfolding $\sigma_{AODV}_\text{def}$ $\sigma_{AODV}'_\text{def}$

proof (rule equalityD1)

show "\{ (f i, p). (\sigma i, \zeta) \in \{ (f i, p) \} \land (\forall j. j \neq i \rightarrow \sigma j \in \text{fst} ' \ \{ (f j, p) \}) \} = \{ (f, p) \}"

by (rule set_eqI) auto

qed

thus "\{ (\sigma, \zeta) | \sigma \zeta s. s \in \text{init} (paodv i) \land (\sigma i, \zeta) = \text{id} s \land (\forall j. j \neq i \rightarrow \sigma j \in (\text{fst o id}) ' \ \text{init} (paodv j)) \} \subseteq \text{init} (opaodv i)"

by simp

next

show "\forall j. \text{init} (paodv j) \neq {}"

unfolding $\sigma_{AODV}_\text{def}$ by simp

next

fix i s a s' \sigma \sigma'

assume "\sigma i = \text{fst} (\text{id} s)"

and "\sigma' i = \text{fst} (\text{id} s')"

and "(s, a, s') \in \text{trans} (paodv i)"

then obtain q q' where "s = (\sigma i, q)"

and "s' = (\sigma' i, q')"

and "((\sigma i, q), a, (\sigma' i, q')) \in \text{trans} (paodv i)"

by (cases s, cases s') auto

from this(3) have ""((\sigma, q), a, (\sigma', q')) \in \text{trans} (opaodv i)"

by simp (rule open_seqp_action [OF aodv_wf])

with \$: s = (\sigma i, q) \land s' = (\sigma' i, q') \$

show ""((\sigma, \text{snd} (\text{id} s)), a, (\sigma', \text{snd} (\text{id} s'))) \in \text{trans} (opaodv i)"

by simp

qed

then interpret op: openproc paodv opaodv id .

have [simp]: "\forall i. (\text{SOME} x. x \in (\text{fst o id}) ' \ \text{init} (paodv i)) = \text{aodv_init} i"

unfolding $\sigma_{AODV}_\text{def}$ by simp

hence "\forall i. \text{openproc.initmissing paodv id i} = \text{initmissing i}"

unfolding op.initmissing_def op.someinit_def initmissing_def

by (auto split: option.split)

thus "\text{openproc.initmissing paodv id} = \text{initmissing} .." .

qed

interpretation aodv_openproc_par_qmsg: openproc_parq paodv opaodv id qmsg

where "aodv_openproc_par_qmsg.netglobal = netglobal"

and "aodv_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: "\forall \sigma. \text{openproc.initmissing} (\lambda i. \text{paodv i} (\langle qmsg \rangle) (\lambda (p, q). (\text{fst} (\text{id} p), \text{snd} (\text{id} p), q)) \ \sigma"
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. paodv i ⟨⟨ qmsg) (λ(p, q). (fst (id p), snd (id p), q)) = initmissing"

by (rule ext)

have "\(P \sigma\). openproc.netglobal (λi. paodv i ⟨⟨ qmsg) (λ(p, q). (fst (id p), snd (id p), q)) 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_aodv_init_netlift
[symmetric, unfolded initmissing_def]

thus "openproc.netglobal (λi. paodv i ⟨⟨ qmsg) (λ(p, q). (fst (id p), snd (id p), q)) = netglobal"

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

thus "openproc.netglobal (λi. paodv 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. paodv i ⟨⟨ qmsg) n)) ⊑ netglobal
(λσ. ∀ i ∈ net_tree_ips n. ∀ dip. irrefl ((rt_graph σ dip) +) )"

(is "_ ⊑ netglobal (λσ. ∀ i. ?inv σ i)")

proof -

from ⟨wf_net_tree n⟩ have proto: "closed (pnet (λi. paodv 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)))"

(is "_ ⊑ netglobal (λσ. ∀ i. ?inv σ i))")

thus "?inv (fst (initmissing (netgmap fst σ))) i"

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

fix σ i

assume sr: "σ ∈ reachable (closed (pnet (λi. paodv i ⟨⟨ qmsg) n)) TT"

hence "∀ i ∈ net_tree_ips n. ?inv (fst (initmissing (netgmap fst σ))) i"

by - (drule invariantD [OF proto],
simp only: aodv_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. paodv 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 = aodv_init i"

by simp

thus ?thesis by simp

qed metis

qed

1.11.5 Loop freedom of AODV

theorem aodv_loop_freedom:

assumes "wf_net_tree n"

shows "closed (pnet (λi. paodv i ⟨⟨ qmsg) n)) ⊑ netglobal (λσ. ∀ dip. irrefl ((rt_graph σ dip) +))"

using assms by (rule aodv_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 ⇒ sqn"
where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ 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 ≠ Suc 0"
  unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 1"
  by simp

2.1.2 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 × k × f × nat × ip × ip set"
definition proj2 :: "r ⇒ sqn" ("\pi_2")
where \( \pi_2 \equiv \lambda(dsn, _, _, _, _, _). dsn \)

definition proj3 :: "r ⇒ k" ("\( \pi_3 \)"")
  where \( \pi_3 \equiv \lambda(_, dsk, _, _, _, _). dsk \)

definition proj4 :: "r ⇒ f" ("\( \pi_4 \)"")
  where \( \pi_4 \equiv \lambda(_, _, flag, _, _, _). flag \)

definition proj5 :: "r ⇒ nat" ("\( \pi_5 \)"")
  where \( \pi_5 \equiv \lambda(_, _, _, hops, _, _). hops \)

definition proj6 :: "r ⇒ ip" ("\( \pi_6 \)"")
  where \( \pi_6 \equiv \lambda(_, _, _, _, nhip, _). nhip \)

definition proj7 :: "r ⇒ ip set" ("\( \pi_7 \)"")
  where \( \pi_7 \equiv \lambda(_, _, _, _, _, pre). pre \)

lemma projs [simp]:
  "\( \pi_2 \)(dsn, dsk, flag, hops, nhip, pre) = dsn"
  "\( \pi_3 \)(dsn, dsk, flag, hops, nhip, pre) = dsk"
  "\( \pi_4 \)(dsn, dsk, flag, hops, nhip, pre) = flag"
  "\( \pi_5 \)(dsn, dsk, flag, hops, nhip, pre) = hops"
  "\( \pi_6 \)(dsn, dsk, flag, hops, nhip, pre) = nhip"
  "\( \pi_7 \)(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 (\( \pi_3 \) x)"
  by (rule k.induct)

lemma proj4_pred [intro]: "[ P val; P inv ] ⇒ P (\( \pi_4 \) x)"
  by (rule f.induct)

lemma proj6_pair_snd [simp]:
  fixes dsn' r
  shows "\( \pi_6 \) (dsn', snd (r)) = \( \pi_6 \) (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" ("route '{_, _}'")

translations
  "route(rt, dip)" = "rt dip"

definition sqn :: "rt ⇒ ip ⇒ sqn"
  where "sqn rt dip ≡ case route(rt, dip) of Some r ⇒ \( \pi_2 \)(r) | None ⇒ 0"

definition sqnf :: "rt ⇒ ip ⇒ k"
  where "sqnf rt dip ≡ case route(rt, dip) of Some r ⇒ \( \pi_3 \)(r) | None ⇒ unk"

abbreviation flag :: "rt ⇒ ip ⇒ f"
  where "flag rt dip ≡ map_option \( \pi_4 \) (route(rt, dip))"

abbreviation dhops :: "rt ⇒ ip ⇒ nat"
  where "dhops rt dip ≡ map_option \( \pi_5 \) (route(rt, dip))"

abbreviation nhop :: "rt ⇒ ip ⇒ ip"
  where "nhop rt dip ≡ map_option \( \pi_6 \) (route(rt, dip))"
abbreviation  
  precs :: "rt ⇒ 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 sqn_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

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 ⇒ 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 "π_2(addpre v pre) = π_2(v)"
unfolding addpre_def by (cases v) simp

lemma proj3_addpre:
  fixes v pre
  shows "π_3(addpre v pre) = π_3(v)"
unfolding addpre_def by (cases v) simp

lemma proj4_addpre:
  fixes v pre
  shows "π_4(addpre v pre) = π_4(v)"
unfolding addpre_def by (cases v) simp

lemma proj5_addpre:
  fixes v pre
  shows "π_5(addpre v pre) = π_5(v)"
unfolding addpre_def by (cases v) simp

lemma proj6_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows "π_6(addpre v npre) = π_6(v)"
unfolding addpre_def by (cases v) simp

lemma proj7_addpre:
  fixes dsn dsk flag hops nhip pre npre
  shows "π_7(addpre v npre) = π_7(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)) (σ_route(rt, dip))"
lemma snd_addpre [simp]:
"\(\forall dsn \ dsn' \ v \ pre. (dsn, and(addpre (dsn', v) pre)) = addpre (dsn, v) pre\)"
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 "\(\pi_2(\text{the (the (addpreRT rt ip' npre) ip)) = \pi_2(\text{the (rt ip}))\)"
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 "\(\pi_3(\text{the (the (addpreRT rt ip' npre) ip)) = \pi_3(\text{the (rt ip}))\)"
using assms [THEN kD_Some] unfolding addpreRT_def by clarsimp

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

lemma flag_addpreRT [simp]:
fixes rt pre ip dip
assumes "dip \in \kD\ rt"
shows "\(\text{flag (the (addpreRT rt dip pre)) ip = flag rt ip} \)"
unfolding addpreRT_def by auto

lemma kD_addpreRT [simp]:
fixes rt dip npre
assumes "dip \in \kD\ rt"
shows "\(\kD(\text{the (addpreRT rt dip npre)) = \kD\ rt} \)"
unfolding kD_def addpreRT_def by clarsimp blast

lemma vD_addpreRT [simp]:
fixes rt dip npre
assumes "dip \in \kD\ rt"
shows "\(\text{vD (the (addpreRT rt dip npre)) = vD rt} \)"
unfolding vD_def addpreRT_def by clarsimp auto

lemma iD_addpreRT [simp]:
fixes rt dip npre
assumes "dip \in \kD\ rt"
shows "\(\text{iD (the (addpreRT rt dip npre)) = iD rt} \)"
unfolding iD_def addpreRT_def by clarsimp auto

lemma nhop_addpreRT [simp]:
fixes rt pre ip dip
assumes "dip \in \kD\ rt"
shows "\(\text{nhop (the (addpreRT rt dip pre)) ip = nhop rt ip} \)"
unfolding sqn_def addpreRT_def by clarsimp

lemma sqn_addpreRT [simp]:
fixes rt pre ip dip
assumes "dip \in \kD\ rt"
shows "\(\text{sqn (the (addpreRT rt dip pre)) ip = sqn rt ip} \)"
unfolding sqn_def addpreRT_def by clarsimp
lemma dhops_addpreRT [simp]:
  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]:
  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]:
  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]:
  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]:
  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]:
  "∀ n hops nhip pre. Suc 0 ≤ n → update_arg_wf (n, kno, val, hops, nhip, pre)"
  unfolding update_arg_wf_def by auto

lemma wf_r_cases [intro]:
  assumes "update_arg_wf r"
  and cl1: "∀ nhip pre. P (0, unk, val, Suc 0, nhip, pre)"
  and cl2: "∀ dsn hops nhip pre. dsn > 0 → 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 r s r t ≡ case σroute(r, s) of
  None ⇒ r (s ↦ r)
  | Some s ⇒
    if π2(s) < π2(r) then r (s ↦ addpre r (π7(s)))
    else if π2(s) = π2(r) ∧ (π5(s) > π5(r) ∨ π4(s) = inv)
      then r (s ↦ addpre r (π7(s)))
      else if π3(r) = unk
        then r (s ↦ addpre (π7(r)))
        else r (s ↦ addpre s (π7(r)))"

lemma update_simps [simp]:
  fixes r s nrt nr nr' ns rt ip
  defines "s ≡ the σroute(r, s)"
  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(r)] ⇒ update r s r t = r (s ↦ r)"" update rt ip r = rt (ip ↦ r)"
  "[ip ∈ kD(r); sqn rt ip < π2(r)] ⇒ update r s r t = r (s ↦ nr)"" update rt ip r = rt (ip ↦ nr)"
  "[ip ∈ kD(r); sqn rt ip = π2(r); the (dhops rt ip) > π5(r)] ⇒ update r s r t = r (s ↦ nr)"" update rt ip r = rt (ip ↦ nr)"
  "[ip ∈ kD(r); sqn rt ip = π2(r); flag rt ip = Some inv] ⇒ update r s r t = r (s ↦ nr)"" update rt ip r = rt (ip ↦ nr)"
  "[ip ∈ kD(r); π3(r) = unk; (π2(r) = 0) = (π3(r) = unk)] ⇒ update r s r t = r (s ↦ nr')"" update rt ip r = rt (ip ↦ nr')"
  "[ip ∈ kD(r); 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 r s r t = r (s ↦ ns)"
proof -
  assume "ip /∈ kD(r)"
  hence "σroute(r, ip) = None" ..
  thus "update rt ip r = rt (ip ↦ r)"
  unfolding update_def by simp
next
  assume "ip ∈ kD(r)"
  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(r)"
and "sqn rt ip = π_2(r)"
and "the (dhops rt ip) > π_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 = π_2(r)⟩ and ⟨the (dhops rt ip) > π_5(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 = π_3(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 = π_2(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 "π_3(r) = unk"
and "(π_2(r) = 0) = (π_3(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 ⟨π_2(r) = 0⟩ and ⟨π_3(r) = unk⟩
show "update rt ip r = rt (ip ↦ nr')"
unfolding update_def nr'_def nr_def s_def by auto
next
assume "ip ∈ kD(rt)"
and otherassms: "sqn rt ip ≥ π_2(r)"
"π_3(r) = kno"
"sqn rt ip = π_2(r) ⟹ the (dhops rt ip) ≤ π_5(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 "(π_2(r) = 0) = (π_3(r) = unk)"
and c1: "[| ip ∉ kD(rt) |] ⟹ P (rt (ip ↦ r))"
and c2: "[| ip ∈ kD(rt); sqn rt ip < π_2(r) |]
⟹ P (rt (ip ↦ addpre r (π_7(\sigma_{route}(rt, ip))))))"
and c3: "[| ip ∈ kD(rt); sqn rt ip = π_2(r); the (dhops rt ip) > π_5(r) |]
⟹ P (rt (ip ↦ addpre r (π_7(\sigma_{route}(rt, ip))))))"
and c4: "[| ip ∈ kD(rt); sqn rt ip = π_2(r); the (flag rt ip) = inv |]
⟹ P (rt (ip ↦ addpre r (π_7(\sigma_{route}(rt, ip))))))"
and c5: "[| ip ∈ kD(rt); π_3(r) = unk |]
⟹ P (rt (ip ↦ (π_7(\sigma_{route}(rt, ip)), π_3(r),
π_4(r), π_5(r), π_6(r), π_7(addpre r (π_7(\sigma_{route}(rt, ip))))))))"
and c6: "[| ip ∈ kD(rt); sqn rt ip ≥ π_2(r); π_3(r) = kno;
sqn rt ip = π_2(r) ⟹ the (dhops rt ip) ≤ π_5(r) ∧ the (flag rt ip) = val |]
⟹ P (rt (ip ↦ addpre (\sigma_{route}(rt, ip)) (π_7(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 \(\text{metis kD\_Some}\)

moreover obtain \(dsn'\) \(dsk'\) \(fl'\) \(hops'\) \(nhip'\) \(pre'\)
where \(req: "r = (dsn', dsk', fl', hops', nhip', pre')"
by \(\text{cases r}\) \(\text{metis}\)

ultimately show ?thesis
using \(\langle\pi_2(r) = 0\rangle \Rightarrow \langle\pi_3(r) = \text{unk}\rangle\)
\[c_2 \ [\text{OF } \langle\text{ip }\in\text{kD\_rt}\rangle]\]
\[c_3 \ [\text{OF } \langle\text{ip }\in\text{kD\_rt}\rangle]\]
\[c_4 \ [\text{OF } \langle\text{ip }\in\text{kD\_rt}\rangle]\]
\[c_5 \ [\text{OF } \langle\text{ip }\in\text{kD\_rt}\rangle]\]
\[c_6 \ [\text{OF } \langle\text{ip }\in\text{kD\_rt}\rangle]\]

unfolding update\_def sqn\_def by auto
qed

lemma update\_cases\_kD:
assumes "\((\pi_2(r) = 0) = (\pi_3(r) = \text{unk})\)"
and "ip \in \text{kD\_rt}\"
and \(c_2: \text{sqn\_rt\_ip < \pi_2(r) = } \Rightarrow \text{P (rt (ip } \mapsto \text{addpre (the \_route (rt, ip)))}})\)
and \(c_3: \text{sqn\_rt\_ip = \pi_2(r); the (dhops\_rt\_ip) >} \pi_5(r)\]
\[\Rightarrow \text{P (rt (ip } \mapsto \text{addpre (the \_route (rt, ip)))}})\)
and \(c_4: \text{sqn\_rt\_ip = \pi_2(r); the (flag\_rt\_ip) = inv}\]
\[\Rightarrow \text{P (rt (ip } \mapsto \text{addpre (the \_route (rt, ip)))}})\)
and \(c_5: \pi_3(r) = \text{unk} \Rightarrow \text{P (rt (ip } \mapsto (\pi_2\_the \_route (rt, ip), \pi_3(r), \pi_4(r), \pi_5(r), \pi_6(r), \pi_7\_addpre\_r (\pi_7\_the \_route (rt, ip))))})\)
and \(c_6: \text{sqn\_rt\_ip } \geq \pi_2(r); \pi_3(r) = \text{kno}; \text{sqn\_rt\_ip = \pi_2(r) } \Rightarrow \text{the (dhops\_rt\_ip) } \leq \pi_5(r) \wedge \text{the (flag\_rt\_ip) = val}\]
\[\Rightarrow \text{P (rt (ip } \mapsto \text{addpre (the \_route (rt, ip))) } (\pi_7(r)))\]

shows "\(\text{P (update rt ip r)}\)"
using asssms(1) proof \(\text{rule update\_cases}\)
assume "sqn\_rt\_ip < \pi_2(r)"
thus "\text{P (rt(ip } \mapsto \text{addpre r (\pi_7 (the (rt ip)))})" by \(\text{rule c2}\)
next
assume "sqn\_rt\_ip = \pi_2(r)"
and "the (dhops\_rt\_ip) > \pi_5(r)"
thus "\text{P (rt(ip } \mapsto \text{addpre r (\pi_7 (the (rt ip)))})"
by \(\text{rule c3}\)
next
assume "sqn\_rt\_ip = \pi_2(r)"
and "the (flag\_rt\_ip) = inv"
thus "\text{P (rt(ip } \mapsto \text{addpre r (\pi_7 (the (rt ip)))})"
by \(\text{rule c4}\)
next
assume "\pi_3(r) = \text{unk}"
thus "\text{P (rt (ip } \mapsto (\pi_2\_the \_route (rt, ip), \pi_3(r), \pi_4(r), \pi_5(r), \pi_6(r), \pi_7\_addpre\_r (\pi_7\_the \_route (rt, ip))))})"
by \(\text{rule c5}\)
next
assume "sqn\_rt\_ip \geq \pi_2(r)"
and "\pi_3(r) = \text{kno}"
and "sqn\_rt\_ip = \pi_2(r) \Rightarrow \text{the (dhops\_rt\_ip) } \leq \pi_5(r) \wedge \text{the (flag\_rt\_ip) = val}\"
thus "\text{P (rt (ip } \mapsto \text{addpre (the \_route (rt, ip))) } (\pi_7(r)))\"
by \(\text{rule c6}\)
qed \(\text{simp} \_\text{add: } \langle\text{ip }\in\text{kD\_rt}\rangle)\)

lemma in\_kD\_after\_update [simp]:
fixes rt nip dsn dsk flag hops nhip pre
shows "\text{kD (update rt nip (dsn, dsk, flag, hops, nhip, pre)) } = \text{insert nip (kD\_rt)}"
unfolding update\_def
by \(\text{cases "rt nip"}\) \text{auto}

lemma nhop\_of\_update [simp]:

179
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

lemma sqn_if_updated:
  fixes rip v rt ip
  shows "sqn (λ x. 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 dan 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 dan 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.split option.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 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.
  181
\[ rt \neq \text{update } rt \text{ dip (dsn, dsk, flg, hops, sip, \{\})} \Rightarrow \pi_3(\text{the (update } rt \text{ dip (dsn, dsk, flg, hops, sip, \{\}) dip})) = dsk\]

Unfolding \text{update_def sqnf_def}

By (clarsimp split: option.splits split_if_asm) auto

Lemma nhop_update_changed_kno_val [simp]: 
\[ rt \neq \text{update } rt \text{ ip (dsn, kno, val, hops, nhip, \{\})} \Rightarrow \text{the (nhop (update } rt \text{ ip (dsn, kno, val, jumps, nhip, \{\}) ip) = nhip} \]

Using assms unfolding update_def

By (clarsimp split: option.split_asm option.split split_if_asm) auto

Lemma flag_update [simp]: 
\[ rt \neq \text{update } rt \text{ dip (dsn, kno, flg, hops, sip, \{\})} \Rightarrow \text{the (flag (update } rt \text{ dip (dsn, kno, flg, hops, sip, \{\}) dip) = flg} \]

Unfolding update_def

By (clarsimp split: option.split_if_asm) auto

Lemma the_flag_Some [dest!]:

fixes ip rt

Assumes "the (flag rt ip) = x"

And "ip \in 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 = \text{update } rt \text{ dip (dsn, dsk, flag, hops, nhip, pre)}"

Shows "\text{dip} \in kD(rt)"

Proof -

Have "\text{dip} \in kD(update rt dip (dsn, dsk, flag, hops, nhip, pre))" by simp with assms show ?thesis by simp

Qed

Lemma nhop_update [simp]: 
\[ rt \neq \text{update } rt \text{ dip (dsn, dsk, flg, hops, sip, \{\})} \Rightarrow \text{the (nhop (update } rt \text{ dip (dsn, dsk, flg, hops, sip, \{\}) dip) = sip} \]

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 "\text{ip} \neq \text{dip}"

Shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqn rt ip"

Using assms unfolding update_def sqnf_def

By (clarsimp split: option.splits) auto

Lemma sqnf_update_another [simp]:

fixes dip ip rt dsn dsk flag hops nhip pre

Assumes "\text{ip} \neq \text{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]:

\[ \forall dip \in vD(update rt dip' (dsn, dsk, val, hops, nhip, pre)) \Rightarrow (dip \in vD(rt) \land dip=dip') \]

Unfolding update_def vD_def by (clarsimp split: option.split_asm split_if_asm)

Invalidating route entries

definition invalidate :: "rt \Rightarrow (ip \Rightarrow sqn) \Rightarrow rt"

Where "invalidate rt dests \equiv \lambda ip. case (rt ip, dests ip) of (None, _) \Rightarrow None"
\begin{enumerate}
\item \((\text{Some } s, \text{None}) \Rightarrow \text{Some } s\)
\item \((\text{Some } (\_, \text{dsk}, \_, \text{hops}, \text{nhip}, \text{pre}), \text{Some } \text{rsn}) \Rightarrow \text{Some (rsn, dsk, inv, hops, nhip, pre)})\)
\end{enumerate}

lemma \text{proj3\_invalidate} [simp]:
\[
\forall \text{dip}. \ \pi_3(\text{the } (\text{invalidate } \text{rt dests} \text{ dip})) = \pi_3(\text{the } (\text{rt dip}))
\]
unfolding \text{invalidate\_def} by (clarsimp split: option.split)

lemma \text{proj5\_invalidate} [simp]:
\[
\forall \text{dip}. \ \pi_5(\text{the } (\text{invalidate } \text{rt dests} \text{ dip})) = \pi_5(\text{the } (\text{rt dip}))
\]
unfolding \text{invalidate\_def} by (clarsimp split: option.split)

lemma \text{proj6\_invalidate} [simp]:
\[
\forall \text{dip}. \ \pi_6(\text{the } (\text{invalidate } \text{rt dests} \text{ dip})) = \pi_6(\text{the } (\text{rt dip}))
\]
unfolding \text{invalidate\_def} by (clarsimp split: option.split)

lemma \text{proj7\_invalidate} [simp]:
\[
\forall \text{dip}. \ \pi_7(\text{the } (\text{invalidate } \text{rt dests} \text{ dip})) = \pi_7(\text{the } (\text{rt dip}))
\]
unfolding \text{invalidate\_def} by (clarsimp split: option.split)

lemma \text{invalidate\_kd\_inv} [simp]:
\[
\forall \text{rt dests}. \ kD (\text{invalidate } \text{rt dests}) = kD \text{ rt}
\]
unfolding \text{invalidate\_def} \text{kD\_def}
by (simp split: option.split)

lemma \text{invalidate\_sqn}:
\begin{align*}
\text{fixes } & \text{rt dip dests} \\
\text{assumes } & \forall \text{rsn}. \ \text{dests dip = Some } \text{rsn} \rightarrow \text{sqn rt dip} \leq \text{rsn} \\
\text{shows } & \text{sqn rt dip} \leq \text{sqn } (\text{invalidate } \text{rt dests} \text{ dip})
\end{align*}
proof (cases \text{"dip} \notin kD(\text{rt})")
\begin{itemize}
\item assume \text{"\neg dip} \notin kD(\text{rt})"
\item hence \text{"dip} \notin kD(\text{rt})" by simp
\end{itemize}
then obtain \text{dsn dsk flag hops nhip pre} where \text{"rt dip = Some } (\text{dsn, dsk, flag, hops, nhip, pre})
by (metis \text{kD\_Some})
with \text{assms} show \text{"sqn rt dip} \leq \text{sqn } (\text{invalidate } \text{rt dests} \text{ dip})
by (cases \text{"dests dip"}) (auto simp add: \text{invalidate\_def} \text{ sqn\_def})
qed simp

lemma \text{sqn\_invalidate\_in\_dests} [simp]:
\begin{align*}
\text{fixes } & \text{dests ipa rsn rt} \\
\text{assumes } & \text{"dests ipa = Some } \text{rsn} \\
& \text{and } \text{"ipa} \in kD(\text{rt})"
\end{align*}
\begin{itemize}
\item shows \text{"sqn } (\text{invalidate } \text{rt dests}) \text{ ipa} = \text{rsn}"
\end{itemize}
unfolding \text{invalidate\_def} \text{ sqn\_def}
using \text{assms}(1) \text{ assms}(2) \text{ [THEN } \text{kD\_Some]}
by clarsimp

lemma \text{dhops\_invalidate} [simp]:
\[
\forall \text{dip}. \ \text{the } (\text{dhops } (\text{invalidate } \text{rt dests} \text{ dip})) = \text{the } (\text{dhops } \text{rt dip})
\]
unfolding \text{invalidate\_def} by (clarsimp split: option.split)

lemma \text{sqnf\_invalidate} [simp]:
\[
\forall \text{dip}. \ \text{sqnf } (\text{invalidate } (\text{rt } \xi) (\text{dests } \xi)) \text{ dip} = \text{sqnf } (\text{rt } \xi) \text{ dip}
\]
unfolding \text{sqnf\_def} \text{ invalidate\_def} by (clarsimp split: option.split)

lemma \text{nhop\_invalidate} [simp]:
\[
\forall \text{dip}. \ \text{the } (\text{nhop } (\text{invalidate } (\text{rt } \xi) (\text{dests } \xi)) \text{ dip}) = \text{the } (\text{nhop } (\text{rt } \xi) \text{ dip})
\]
unfolding \text{invalidate\_def} by (clarsimp split: option.split)

lemma \text{invalidate\_other} [simp]:
\begin{itemize}
\item fixes \text{rt dests dip}
\item assumes \text{"dip} \notin \text{dom(\text{dests})}"
\item shows \text{"invalidate rt dests dip} = \text{rt dip}"
\end{itemize}
using \text{assms} unfolding \text{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" ("σ_queue(_ _)")
where "σ_queue(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]))"
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" ("σ_p-flag'(_, _, _)")
  where "σ_p-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)"

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. σ_p-flag(store, dip) is partial.
4. The routing table (rt) is modelled as a map (ip ⇒ r 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 ⇒ sqn" ip
  | Newpkt data ip
  | Pkt data ip ip

instance by intro_classes (simp add: eq_newpkt_def)
The \textit{msg} type models the different messages used within AODV. The instantiation as a \textit{msg} is a technicality due to the special treatment of \textit{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.

\begin{verbatim}
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).
        \hspace{1em} 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 \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{verbatim}

\section{The AODV protocol}

theory B_Aodv
imports B_Aodv_Data B_Aodv_Message
  "../../../AWN/AWN_SOS_Labels" "../../../AWN/AWN_Invariants"
begin

\subsection{Data state}

record state =
  ip :: "ip"
  sn :: "sqn"
  rt :: "rt"
  rreqs :: "(ip \times rreqid) set"
  store :: "store"

end
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),
  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"
2.3.2 Auxiliary 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' ∧
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"
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 =
  PAodv
| PNewPkt
| PPkt
| PRreq
| PRrep
| PRerr

fun nat_of_seqp :: "pseqp ⇒ 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) = 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 Γ_AODV :: "("state, msg, pseqp, pseqp label) seqp_env"
where
"Γ_AODV PAodv = labelled PAodv (receive(λmsg'. ξ. (| msg := msg' |)).
 (| is_newpkt | NEWPKT(λξ. (data ξ, ip ξ))
 ⊕ (is_pkt) | PKT(λξ. (data ξ, dip ξ, oip ξ))
 ⊕ (is_rreq)
 [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. (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))"

190
\(\lambda\). \{\xi\} \{\xi\} dip := dip \} | dip. dip \in qD(\text{store} \xi) \wedge vD(\text{rt} \xi)\}
\[\{\xi. \xi \{\text{store} := \text{hd}(\text{queue}(\text{store} \xi, \text{dip} \xi)) \}\}
\unicast(\lambda\xi. \text{the (nhop (\text{rt} \xi) (dip \xi)), } \lambda\xi. \text{pkt(data} \xi, \text{dip} \xi, \text{ip} \xi))\).
\[\{\xi. \xi \{\text{store} := \text{the (drop (dip \xi) (store} \xi)) \}\}
\text{AODV()}\)
\[\{\xi. \xi \{\text{dests} := (\lambda\text{rreqs} \in vD (\text{rt} \xi) \wedge \text{nhop (\text{rt} \xi) rip} = \text{nhop (\text{rt} \xi) (dip} \xi))
\text{then Some (inc (sqn (\text{rt} \xi) \text{rip})) else None) \}\}
\text{rt} \xi \{\text{invalidate (dip} \xi) \} \}
\[\{\xi. \xi \{\text{store} := \text{unsetRFF (store} \xi) \} \}
\[\{\xi. \xi \{\text{rreqid} := \text{nrreqid (rreqs} \xi) \} \}
\[\{\xi. \xi \{\text{rreqs} := \text{rreqs} \cup \{(\text{ip} \xi, \text{rreqid} \xi)\} \}
\text{broadcast(\lambda\xi. \text{rreq(0, rreqid} \xi, \text{dip} \xi, \text{sqn (rt} \xi) (dip} \xi), \text{sqnf (rt} \xi) (dip} \xi), \text{ip} \xi, \text{sn} \xi, \text{ip} \xi)). \text{AODV()}}\)
\[\text{G}_\text{AODV} \text{ PNewPkt = labelled PNewPkt (}
\{\xi. \xi \{\text{dip} \xi = \text{ip} \xi\}
\text{deliver(\lambda\xi. data} \xi). \text{AODV()}}\)
\[\{\xi. \xi \{\text{dip} \xi \neq \text{ip} \xi\}
\text{AODV()}}\]
\[\text{G}_\text{AODV} \text{ PPkt = labelled PPkt (}
\{\xi. \xi \{\text{dip} \xi = \text{ip} \xi\}
\text{deliver(\lambda\xi. data} \xi). \text{AODV()}}\)
\[\{\xi. \xi \{\text{dip} \xi \neq \text{ip} \xi\}
\text{AODV()}}\]
\[\text{G}_\text{AODV} \text{ PRreq = labelled PRreq (}
\{\xi. \{\text{oip} \xi, \text{rreqid} \xi \in \text{rreqs} \xi\}
\text{AODV()}}\)
\[\{\xi. \{\text{oip} \xi, \text{rreqid} \xi \notin \text{rreqs} \xi\}
\text{AODV()}}\]
\[\{\xi. \xi \{\text{rt} := \text{update (rt} \xi) (\text{oip} \xi) \} (\text{osn} \xi, \text{kno}, \text{val}, \text{hops} \xi + 1, \text{sip} \xi, \{\}) \}
\text{rreqs := rreqs} \xi \cup \{(\text{oip} \xi, \text{rreqid} \xi)\} \} \]
\( \text{PRrep} = \text{labelled PRrep} \)

\[
\Gamma_{\text{AODV}} \quad \begin{array}{l}
\text{unicast} (\lambda \xi. \text{the (nhop (rt \xi) (oip \xi))), \lambda \xi. \text{rrep (0, dip \xi, sn \xi, oip \xi, ip \xi))}. \text{AODV}() \\
\end{array}
\]

\[
\begin{array}{l}
\text{dests := (\lambda \xi. \text{if (ip} (\xi) (\text{nhop (rt (rt \xi) \pi) rip = nhop (rt (rt \xi) (oip \xi))})} \\
\text{then Some (inc (sqn (rt (rt \xi) rip)) else None))} \) \\
\end{array}
\]

\[
\begin{array}{l}
\text{rt := invalidate (rt \xi) (dests \xi))} \\
\text{store := setRRT (store \xi) (dests \xi))} \\
\text{pre := \{ the (precs (rt \xi) rip) \} \cup rip. rip \in \text{dom (dests \xi)} \}} \\
\text{dests := (\lambda \xi. \text{if (dests \xi) rip \neq None \wedge the (precs (rt \xi) rip) \neq \} \\
\text{then (dests \xi) rip else None})} \) \\
\end{array}
\]

\[
\text{unicast} (\lambda \xi. \text{the (nhop (rt \xi) (oip \xi))), \lambda \xi. \text{rrep (the (dhops (rt \xi) (dip \xi)), dip \xi, sqn (rt \xi) (dip \xi), oip \xi, ip \xi))}. \text{AODV}() \\
\]

\[
\begin{array}{l}
\text{dests := (\lambda \xi. \text{if (ip} (\xi) (\text{nhop (rt (rt \xi) \pi) rip = nhop (rt (rt \xi) (oip \xi))})} \\
\text{then Some (inc (sqn (rt (rt \xi) rip)) else None))} \) \\
\end{array}
\]

\[
\begin{array}{l}
\text{rt := invalidate (rt \xi) (dests \xi))} \\
\text{store := setRRT (store \xi) (dests \xi))} \\
\text{pre := \{ the (precs (rt \xi) rip) \} \cup rip. rip \in \text{dom (dests \xi)} \}} \\
\text{dests := (\lambda \xi. \text{if (dests \xi) rip \neq None \wedge the (precs (rt \xi) rip) \neq \} \\
\text{then (dests \xi) rip else None})} \) \\
\end{array}
\]

\[
\text{unicast} (\lambda \xi. \text{the (nhop (rt \xi) (oip \xi))), \lambda \xi. \text{rrep (the (dhops (rt \xi) (dip \xi)), dip \xi, sqn (rt \xi) (dip \xi), oip \xi, ip \xi))}. \text{AODV}() \\
\]

\[
\begin{array}{l}
\text{dests := (\lambda \xi. \text{if (ip} (\xi) (\text{nhop (rt (rt \xi) \pi) rip = nhop (rt (rt \xi) (oip \xi))})} \\
\text{then Some (inc (sqn (rt (rt \xi) rip)) else None))} \) \\
\end{array}
\]

\[
\begin{array}{l}
\text{rt := invalidate (rt \xi) (dests \xi))} \\
\text{store := setRRT (store \xi) (dests \xi))} \\
\text{pre := \{ the (precs (rt \xi) rip) \} \cup rip. rip \in \text{dom (dests \xi)} \}} \\
\text{dests := (\lambda \xi. \text{if (dests \xi) rip \neq None \wedge the (precs (rt \xi) rip) \neq \} \\
\text{then (dests \xi) rip else None})} \) \\
\end{array}
\]

\[
\text{unicast} (\lambda \xi. \text{the (nhop (rt \xi) (oip \xi))), \lambda \xi. \text{rrep (the (dhops (rt \xi) (dip \xi)), dip \xi, sqn (rt \xi) (dip \xi), oip \xi, ip \xi))}. \text{AODV}() \\
\]

\[
\begin{array}{l}
\text{dests := (\lambda \xi. \text{if (ip} (\xi) (\text{nhop (rt (rt \xi) \pi) rip = nhop (rt (rt \xi) (oip \xi))})} \\
\text{then Some (inc (sqn (rt (rt \xi) rip)) else None))} \) \\
\end{array}
\]

\[
\begin{array}{l}
\text{rt := invalidate (rt \xi) (dests \xi))} \\
\text{store := setRRT (store \xi) (dests \xi))} \\
\text{pre := \{ the (precs (rt \xi) rip) \} \cup rip. rip \in \text{dom (dests \xi)} \}} \\
\text{dests := (\lambda \xi. \text{if (dests \xi) rip \neq None \wedge the (precs (rt \xi) rip) \neq \} \\
\text{then (dests \xi) rip else None})} \) \\
\end{array}
\]

\[
\text{unicast} (\lambda \xi. \text{the (nhop (rt \xi) (oip \xi))), \lambda \xi. \text{rrep (the (dhops (rt \xi) (dip \xi)), dip \xi, sqn (rt \xi) (dip \xi), oip \xi, ip \xi))}. \text{AODV}() \\
\]

\[
\begin{array}{l}
\text{dests := (\lambda \xi. \text{if (ip} (\xi) (\text{nhop (rt (rt \xi) \pi) rip = nhop (rt (rt \xi) (oip \xi))})} \\
\text{then Some (inc (sqn (rt (rt \xi) rip)) else None))} \) \\
\end{array}
\]

\[
\begin{array}{l}
\text{rt := invalidate (rt \xi) (dests \xi))} \\
\text{store := setRRT (store \xi) (dests \xi))} \\
\text{pre := \{ the (precs (rt \xi) rip) \} \cup rip. rip \in \text{dom (dests \xi)} \}} \\
\text{dests := (\lambda \xi. \text{if (dests \xi) rip \neq None \wedge the (precs (rt \xi) rip) \neq \} \\
\text{then (dests \xi) rip else None})} \) \\
\end{array}
\]
\[ \Gamma_{AODV} \text{PRerr} = \text{labelled PRerr} \]

\[
\begin{align*}
\xi. \xi. & \text{dests} := (\lambda \text{rip. case (dests } \xi) \text{ rip of None } \Rightarrow \text{None}) \\
& \text{if rip } \in \text{vD (rt } \xi) \wedge \text{ the (nhop (rt } \xi) \text{ rip) = sip } \xi \wedge \text{ sqn (rt } \xi) \text{ rip < rsn then Some rsn else None) } \Rightarrow \text{None} \\
\xi. & \text{rt := invalidate (rt } \xi \text{)} \text{ (dests } \xi) \Rightarrow \text{None} \\
\xi. & \text{store := setRRF (store } \xi \text{)} \text{ (dests } \xi) \Rightarrow \text{None} \\
\xi. & \text{pre := } \bigcup \{ \text{ the (precs (rt } \xi) \text{ rip) } | \text{ rip } \in \text{ dom (dests } \xi) \} \Rightarrow \text{None} \\
\xi. & \text{dests} := (\lambda \text{rip. if ((dests } \xi) \text{ rip } \neq \text{ None } \wedge \text{ the (precs (rt } \xi) \text{ rip) } \neq \{\}) \\
& \text{then (dests } \xi \text{ rip else None) } \Rightarrow \text{None} \\
\text{groupcast}(\lambda \xi. \text{pre }, \lambda \xi. \text{rerr(dests } \xi, \text{ip } \xi)). \text{AODV()}
\end{align*}
\]

\[
\begin{align*}
\text{declare } & \Gamma_{AODV}.\text{simps [simp del, code del]} \\
\text{lemmas } & \Gamma_{AODV}_\text{simps [simp, code]} = \Gamma_{AODV}.\text{simps [simplified]}
\end{align*}
\]

\textbf{fun } \Gamma_{AODV}_\text{skeleton where}

\[
\begin{align*}
\text{"} & \Gamma_{AODV}_\text{skeleton PAodv = seqp_skeleton (}\Gamma_{AODV} \text{ PAodv)"
\text{"} & \Gamma_{AODV}_\text{skeleton PNewPkt = seqp_skeleton (}\Gamma_{AODV} \text{ PNewPkt)"
\text{"} & \Gamma_{AODV}_\text{skeleton PPkt = seqp_skeleton (}\Gamma_{AODV} \text{ PPkt)"
\text{"} & \Gamma_{AODV}_\text{skeleton PRreq = seqp_skeleton (}\Gamma_{AODV} \text{ PRreq)"
\text{"} & \Gamma_{AODV}_\text{skeleton PRrep = seqp_skeleton (}\Gamma_{AODV} \text{ PRrep)"
\text{"} & \Gamma_{AODV}_\text{skeleton PRerr = seqp_skeleton (}\Gamma_{AODV} \text{ PRerr)"
\end{align*}
\]

\textbf{lemma } \Gamma_{AODV}_\text{skeleton_wf [simp]}:

"wellformed \Gamma_{AODV}_\text{skeleton}"

\textbf{proof (rule, intro allI)}

\text{fix } pn pn'

\text{show "call(pn') } \not\in \text{ stermso (}\Gamma_{AODV}_\text{skeleton pn)"

\text{by (cases pn) simp_all}

\textbf{qed}

\textbf{declare } \Gamma_{AODV}_\text{skeleton.simps [simp del, code del]}

\textbf{lemmas } \Gamma_{AODV}_\text{skeleton_simps [simp, code]} = \Gamma_{AODV}_\text{skeleton.simps [simplified } \Gamma_{AODV}_\text{simps seqp_skeleton.simps]}

\textbf{lemma } aodv_proc_cases [dest]:

\textbf{fixes } p pn

\textbf{shows } "p } \in \text{ ctermso (}\Gamma_{AODV} \text{ pn} ) \Rightarrow
\begin{align*}
(p } & \in \text{ ctermso (}\Gamma_{AODV} \text{ PAodv) } \vee
p } & \in \text{ ctermso (}\Gamma_{AODV} \text{ PNewPkt) } \vee
p } & \in \text{ ctermso (}\Gamma_{AODV} \text{ PPkt) } \vee
p } & \in \text{ ctermso (}\Gamma_{AODV} \text{ PRreq) } \vee
p } & \in \text{ ctermso (}\Gamma_{AODV} \text{ PRrep) } \vee
p } & \in \text{ ctermso (}\Gamma_{AODV} \text{ PRerr) })
\end{align*}

\textbf{using asssms}

\textbf{by (cases pn) simp_all}

\textbf{definition } \sigma_{AODV} :: "ip } \Rightarrow (\text{state } \times (\text{state, msg, pseqp, pseqp label) seq}) \text{ set}"

\textbf{where } "\sigma_{AODV} i } \equiv \{(\text{aodv_init i, } \Gamma_{AODV} \text{ PAodv)}\"

\textbf{abbreviation } paodv

:: "ip } \Rightarrow (\text{state } \times (\text{state, msg, pseqp, pseqp label) seqp, msg seq_action}) \text{ automaton}"

\textbf{where}

"paodv i } \equiv \{(\text{init } = \sigma_{AODV} i, \text{ trans } = \text{seq_sos } \Gamma_{AODV} )\"

\textbf{lemma } aodv_trans: "\text{trans (paodv i) = seq_sos } \Gamma_{AODV}"

\textbf{by simp}

\textbf{lemma } aodv_control_within [simp]: "\text{control_within } \Gamma_{AODV} \text{ (init (paodv i))}"

\textbf{unfolding } \sigma_{AODV}_\text{def by (rule control_withinI) (auto simp del: } \Gamma_{AODV}_\text{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 "\( \exists \, p \, l \thinspace . \thinspace P \, l \, p = \Rightarrow \, Q \)"
  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 = \Rightarrow \, labels \, \Gamma_{AODV} \, p = \{PAodv-:0\}"
  unfolding \( \sigma_{AODV} \)_def by simp

lemma aodv_init_kD_empty [simp]: "(\( \xi \), p) \in \sigma_{AODV} \, i = \Rightarrow \, kD \, (rt \, \xi) = {}"
  unfolding \( \sigma_{AODV} \)_def kD_def by simp

lemma aodv_init_sip_not_ip [simp]: "\neg (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 (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 [cterm_env]
aodv_proc_cases [ctermml_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 ⇒ bool"
  where "not_Pkt m ≡ case m of Pkt _ _ _ ⇒ False | _ ⇒ True"

definition msg_sender :: "msg ⇒ ip"
  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"
"⋀ d 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 _ _ _ _ oipc osnc ipcc ⇒ hopsc = 0 −→ oipc = 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"
"⋀ 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 hopsc _ _ _ _ oipc 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 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
                       ∧ (the (dhops crt oipc) ≤ hopsc)
                       ∧ (the (flag crt oipc) = val)))
                     | Rrep hopsc dipc _ _ ipcc ⇒ (ipcc ≠ dipc −→ dipc∈kD(crt)
                       ∧ sqn crt dipc = dsnc
                       ∧ (the (dhops crt dipc) = hopsc
                       ∧ (the (flag crt dipc) = val))"
lemma rreq_rrep_fresh [simp]:
  "\hops \rreqid \dsn \dsk \oip \osn \sip. rreq_rrep_fresh \crt (Rreq \hops \rreqid \dip \dsn \dsk \oip \osn \sip) =
    (\sip \neq \oip \rightarrow \oip \in kD(\crt)
     \land (sqn \crt \oip > \osn
     \lor (sqn \crt \oip = \osn
     \land the (dhops \crt \oip) \leq \hops
     \land the (flag \crt \oip) = val))")"

"\hops \dsn \oip \sip. rreq_rrep_fresh \crt (Rrep \hops \dip \dsn \oip \sip) =
    (\sip \neq \dip \rightarrow \dip \in kD(\crt)
     \land sqn \crt \dip = \dsn
     \land the (dhops \crt \dip) = \hops
     \land the (flag \crt \dip) = val)"

"\dests \sip. rreq_rrep_fresh \crt (Rerr \dests \sip) = True"

"\dip. rreq_rrep_fresh \crt (Newpkt \dip) = True"

"\dip \sip. rreq_rrep_fresh \crt (Pkt \dip \sip) = True"

unfolding rreq_rrep_fresh_def by simp_all

definition rerr_invalid :: "rt \Rightarrow msg \Rightarrow bool"
where "rerr_invalid \crt \m = case \m of Rerr \destsc _ \Rightarrow (\\ripc \in dom(\destsc).
    (\ripc \in iD(\crt) \land the (\destsc \ripc) = sqn \crt \ripc))
    \rightarrow True"

lemma rerr_invalid [simp]:
  "\hops \rreqid \dsn \dsk \oip \osn \sip. rerr_invalid \crt (Rreq \hops \rreqid \dip \dsn \dsk \oip \osn \sip) = True"

"\hops \dsn \oip \sip. rerr_invalid \crt (Rrep \hops \dip \dsn \oip \sip) = True"

"\dests \sip. rerr_invalid \crt (Rerr \dests \sip) = (\\ripc \in iD(\crt) \land the (\dests \ripc) = sqn \crt \ripc)"

"\dip. rerr_invalid \crt (Newpkt \dip) = True"

"\dip \sip. rerr_invalid \crt (Pkt \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 aodv_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 (netmap fst \sigma)) i = aodv_init i"
  using assms unfolding initmissing_def by simp

lemma fst_initmissing_netmap_pair_fst [simp]:
  "fst (initmissing (netmap (\lambda (p, q). (fst (id p), snd (id p), q)) \sigma)) =
    fst (initmissing (netmap fst \sigma)))"

unfolding initmissing_def by auto

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

lemma fst_initmissing_netmap_default_aodv_init_netlift:
  "fst (initmissing (netmap fst \sigma)) = default aodv_init (netlift fst \sigma)"

unfolding initmissing_def default_def
by (simp add: fst_netmap_netlift_def: One_net_def)

definition netglobal :: "((nat \Rightarrow state) \Rightarrow bool) \Rightarrow ((state \times 'b) \times 'c) \Rightarrow bool"
where "netglobal \P = (\lambda s. \P (default aodv_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
  nsqn :: "r ⇒ sqn"
where
  "nsqn r ≡ if π₄(r) = val ∨ π₂(r) = 0 then π₂(r) else (π₂(r) - 1)"

lemma nsqn_r_def':
  "nsqn r = (if π₄(r) = inv then π₂(r) - 1 else π₂(r))"
  unfolding nsqn_r_def by simp

lemma nsqn_r_zero [simp]:
  "∀dsn dsk flag hops nhip pre. nsqn (0, dsk, flag, hops, nhip, pre) = 0"
  unfolding nsqn_r_def by clarsimp

lemma nsqn_r_val [simp]:
  "∀dsn dsk hops nhip pre. nsqn (dsn, dsk, val, hops, nhip, pre) = dsn"
  unfolding nsqn_r_def by clarsimp

lemma nsqn_r_inv [simp]:
  "∀dsn dsk hops nhip pre. nsqn (dsn, dsk, inv, hops, nhip, pre) = dsn - 1"
  unfolding nsqn_r_def by clarsimp

lemma nsqn_r_lte_dsn [simp]:
  "∀dsn dsk flag hops nhip pre. nsqn (dsn, dsk, flag, hops, nhip, pre) ≤ dsn"
  unfolding nsqn_r_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 ⇒ 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 (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_rethe (σroute(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]:
lemma sqn_sqn:
"∀ rt dip. sqn rt dip - 1 ≤ sqn rt dip"
unfolding sqn_def sqn_def by (clarsimp split: option.split)

lemma nsqn_sqn: "nsqn rt dip ≤ sqn rt dip"
unfolding sqn_def sqn_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_def update_def
by (clarsimp simp: kD_nsqn split: option.split_asm option.split split_if_asm)
(fmetis 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_def
by (frule kD_Some) (clarsimp simp: 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 simp: 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' ≡ ((nsqn r < nsqn r') ∨ (nsqn r = nsqn 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 "(nsqn r < nsqn r') ∨ (nsqn r = nsqn r' ∧ π₅(r) ≥ π₅(r'))"
  shows "r ⊑ r'"
  unfolding fresher_def
  using assms .

lemma fresherE [elim]:
  assumes "r ⊑ r'" and "nsqn r < nsqn r' ==> P r r'"
  and "nsqn r = nsqn 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

2.5.3 Comparing routing tables

definition
  rt_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
where
  "rt_fresher ≡ λdip rt rt'. (the (σ_route(rt, dip))) ⊑ (the (σ_route(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) = (nsqn r (the (rt1 i)) < nsqn r (the (rt2 i)) ∨ nsqn r (the (rt1 i)) = nsqn r (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" ("(_≈_ _)")
   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) ⊑ the (rt2 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 "π₅(the (rt dip)) = π₅(the (rt' dip))"
  shows "rt ≈ dip rt'"
proof
  from assms(1-2,4) have dhops': "the (dhops rt' dip) ≤ the (dhops rt dip)"
    by (simp add: proj5_eq_dhops)
  with assms(1-3) show "rt ⊑ dip rt'"
    by (rule rt_fresherI2)
next
  from assms(1-2,4) have dhops: "the (dhops rt dip) ≤ the (dhops rt' dip)"
    by (simp add: proj5_eq_dhops)
  with assms(2,1) assms(3) [symmetric] show "rt' ⊑ dip rt"
    by (rule rt_fresherI2)
qed
lemma \texttt{rt\_fresh\_asE [elim]}:
assumes "rt1 \approx dip rt2"
and "[ rt1 \subseteq dip rt2; rt2 \subseteq dip rt1 ] \implies P rt1 rt2 dip"
shows "P rt1 rt2 dip"
using \texttt{assms unfolding rt\_fresh\_as_def by simp}

lemma \texttt{rt\_fresh\_asD1 [dest]}:
assumes "rt1 \approx dip rt2"
shows "rt1 \subseteq dip rt2"
using \texttt{assms unfolding rt\_fresh\_as_def by simp}

lemma \texttt{rt\_fresh\_asD2 [dest]}:
assumes "rt1 \approx dip rt2"
shows "rt2 \subseteq dip rt1"
using \texttt{assms unfolding rt\_fresh\_as_def by simp}

lemma \texttt{rt\_fresh\_as_sym}:
assumes "rt1 \approx dip rt2"
shows "rt2 \approx dip rt1"
using \texttt{assms unfolding rt\_fresh\_as_def by simp}

lemma \texttt{not\_rt\_fresh\_asI1 [intro]}:
assumes "\neg (rt1 \subseteq dip rt2)"
shows "\neg (rt1 \approx dip rt2)"
proof
assume "rt1 \approx dip rt2"
hence "rt1 \subseteq dip rt2" ..
with \(\neg (rt1 \subseteq dip rt2)\) show False ..
qed

lemma \texttt{not\_rt\_fresh\_asI2 [intro]}:
assumes "\neg (rt2 \subseteq dip rt1)"
shows "\neg (rt1 \approx dip rt2)"
proof
assume "rt1 \approx dip rt2"
hence "rt2 \subseteq dip rt1" ..
with \(\neg (rt2 \subseteq dip rt1)\) show False ..
qed

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

lemmas \texttt{not\_single\_rt\_fresher\_asI1 [intro]} = \texttt{not\_rt\_fresh\_asI1 [OF not\_single\_rt\_fresher]}
lemmas \texttt{not\_single\_rt\_fresher\_asI2 [intro]} = \texttt{not\_rt\_fresh\_asI2 [OF not\_single\_rt\_fresher]}

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

lemma \texttt{rt\_fresh\_as\_nsqnr}:
assumes "dip \in kD(rt1)"
and "dip \in kD(rt2)"

\texttt{202}
and "rt1 ≈ dip rt2"
shows "nsqn_r (the (rt2 dip)) = nsqn_r (the (rt1 dip))"
using assms(3) unfolding rt_fresher_as_def
by (auto simp: rt_fresher_def2 [OF ⟨dip ∈ kD(rt1)⟩ ⟨dip ∈ kD(rt2)⟩]
    kD_nsqn [OF ⟨dip ∈ kD(rt2)⟩]
    kD_nsqn [OF ⟨dip ∈ kD(rt1)⟩])

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 "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
  using assms unfolding rt_fresher_update_other
next
  assume "dip = ip" from ⟨dip ∈ kD(rt)⟩ rt have "sqn rt dip = dsn_n"
  and "the (dhops rt dip) = hops_n" and "the (flag rt dip) = f_n"
  by (simp add: sqn_def proj5_eq_dhops [symmetric]
    proj4_eq_flag [symmetric])+
  from ⟨update_arg_wf r⟩ have "("dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)" ⊑ the ((update rt dip r) dip)"
  proof (rule wf_r_cases)
    fix nhip pre
    from ⟨hops_n ≥ 1⟩ have "∀pre'. (dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)" ⊑ (dsn_n, unk, val, Suc 0, nhip, pre')" unfolding fresher_def sqn_def by (cases f_n) auto
    thus "("dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)" ⊑ 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 "("dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)" ⊑ the (update rt dip (dsn, kno, val, hops, nhip, pre) dip)"
proof (rule update_cases_kD [OF _ (dip\in\kD rt)], simp_all add: \(0 < dsn\))
assume "dsn \_ < dsn"
thus "(dsn, dsk, f, hops, nhp, pre) \subseteq (dsn, kno, val, hops, nhp, pre)"
unfolding fresher_def by auto

next
assume "dsn \_ = dsn"
and "hops < hops"
thus "(dsn, dsk, f, hops, nhp, pre) \subseteq (dsn, kno, val, hops, nhp, pre)"
unfolding fresher_def by simp

next
assume "dsn \_ = dsn"
with \(0 < dsn\)
show "(dsn, dsk, f, hops, nhp, pre) \subseteq (dsn, kno, val, hops, nhp, pre)"
unfolding fresher_def by simp

qed

hence "rt \subseteq 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 \in\kD rt"
and indests: "\forall rip \in\dom\dests. rip \in\vD rt \land sqn rt rip < the (dests rip)"
shows "rt \subseteq dip invalidate rt dests"
proof (cases "dip \in\dom\dests")
assume "dip \in\dom\dests"
thus ?thesis using (dip\in\kD rt)
by - (rule single_rt_fresher, simp)

next
assume "dip \in\dom\dests"
moreover with indests
have "dip \in\vD rt"
and "the (dests dip) = inc (sqn rt 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 \in\kD rt"
and "dip \in\dom\dests"
shows "nsqn r (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 \in\kD rt"
and "\forall rip \in\dom\dests. rip \in\vD rt \land the (dests rip) = inc (sqn rt rip)"
shows "rt \approx dip invalidate rt dests"
proof (cases "dip \in\dom\dests")
assume "dip \in\dom\dests"
with (dip\in\kD rt) have "dip\in\kD (invalidate rt dests)"
by simp
with (dip\in\kD rt) show ?thesis
by rule (simp_all add: (dip\in\dom\dests))

next
assume "dip \in\dom\dests"
with assms(2) have "dip \in\vD rt"
and "the (dests dip) = inc (sqn rt dip)" by auto
from (dip\in\vD rt) have "dip\in\kD rt" by simp
moreover then have "dip\in\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 = nsqn_r (the (invalidate rt dests dip))"
proof -
from (dip∈kD(rt)) have "nsqn_r (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 = nsqn_r (the (invalidate rt dests dip))" by simp
qed
also have "sqn rt dip = nsqn_r (the (invalidate rt dests dip))"
proof -
from (dip∈kD(invalidate rt dests)) have "nsqn_r (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 ⊑ dip rt2"
  and "¬(rt2 ⊑ rt1)"
  shows "rt1 ⊏ i rt2"
  using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherE' [elim]:
  assumes "rt1 ⊑ rt2"
  and "[ rt1 ⊑ rt2; ¬(rt2 ⊑ 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 ⊑ rt2"
  and "¬(rt1 ≈ 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 ⊑ rt2"
  and "[ rt1 ⊑ rt2; ¬(rt1 ≈ rt2) ] ⊢ P rt1 rt2 i"
  shows "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 ⊏ rt2 =
  (nsqn_r (the (rt1 i)) < nsqn_r (the (rt2 i))
  ∨ (nsqn_r (the (rt1 i)) = nsqn_r (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 ⊏ rt2"
shows "the (rt1 dip) ⊑ 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 ⊏ rt2"
shows "¬ rt1 ≈ rt2"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
assumes "rt1 ⊏ rt2"
and "rt2 ⊏ rt3"
shows "rt1 ⊏ rt3"
proof -
  from rt1 ⊏ rt2 obtain "the (rt1 dip) ⊑ the (rt2 dip)" by auto
  also from rt2 ⊏ rt3 obtain "the (rt2 dip) ⊑ the (rt3 dip)" by auto
  finally have "the (rt1 dip) ⊑ the (rt3 dip)".
moreover have "¬ (rt1 ≈ rt3)"
proof -
  from rt1 ⊏ rt2 obtain "¬(the (rt2 dip) ⊑ the (rt1 dip))" by auto
  also from rt2 ⊏ rt3 obtain "¬(the (rt3 dip) ⊑ the (rt2 dip))" by auto
  finally have "¬(the (rt3 dip) ⊑ the (rt1 dip))".
  thus ?thesis ..
qed ultimately show "rt1 ⊏ rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: "¬ (rt ⊏ rt)"
using assms unfolding rt_strictly_fresher_def by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
assumes "rt1 ⊏ rt2"
and "rt2 ⊏ rt3"
shows "rt1 ⊏ rt3"
proof -
  from rt1 ⊏ rt2 have "rt1 ⊏ rt2"
  and "¬(rt2 ⊏ rt1)"
  unfolding rt_strictly_fresher_def'' by auto
  from this(1) and rt1 ⊏ rt3 have "rt1 ⊏ rt3" ..
moreover from ¬(rt2 ⊏ rt1) have "¬(rt3 ⊏ rt1)"
proof (rule contrapos_nn)
  assume "rt3 ⊏ rt1"
  with rt2 ⊏ rt3 show "rt2 ⊏ rt1" ..
qed ultimately show "rt1 ⊏ 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

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

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 \( \text{"dip} \in \text{dom(dests)} \)\n
hence \( \text{"dip} \notin \text{vD(invalidate rt1 dests)} \)\n
  unfolding invalidate_def vD_def
  by clarsimp (metis assms option.simps(3) vD_invalidate_vD_not_dests)
with \( \text{"dip} \in \text{vD(invalidate rt1 dests)} \); show \(?thesis \) by simp

next

assume \( \text{"dip} \notin \text{dom(dests)} \)\n
hence \( \text{"dip} = \text{None} \) by auto

moreover with \( \text{\{\text{dip} \in \text{vD(invalidate rt1 dests)}\}} \) have \( \text{\{\text{dip} \in \text{vD(rt1)}\}} \)

  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!]:
  \( \lor \text{\{\text{ip} \neq \text{dip}; rt \sqsubseteq dip rt'\}} \implies \text{update rt ip r} \sqsubseteq dip rt' \)

unfolding rt_strictly_fresher_def' by clarsimp

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

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

lemma dhops_le_hops_imp_update_strictly_fresher:
  assumes \( \text{\{\text{dip} \in \text{vD (rt2 nhip)}\}} \)
  and \( \text{\{sqn (rt2 nhip) dip = osn\}} \)
  and \( \text{\{\text{the (dhops (rt2 nhip) dip) \leq hops\}} \}
  and \( \text{\{\text{rt} \neq \text{update rt dip (osn, kno, val, Suc hops, nhip, \{\})}\}} \)
  shows \( \text{\{\text{update rt dip (osn, kno, val, Suc hops, nhip, \{}) \sqsubseteq dip rt2 nhip\}} \)
unfolding rt_strictly_fresher_def'
proof (rule disjI2, rule conjI)
  from ** have \( \text{\{nsqn (update rt dip (osn, kno, val, Suc hops, nhip, \{)) dip = osn\}} \)
    by (rule nsqn_update_changed_kno_val)
with \( \text{\{dip \in vD(rt2 nhip)\}} \)
  have \( \text{\{nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip, \{)) dip)) = osn\}} \)
    by (simp add: kD_nsqn)
  also have \( \text{\{osn = sqn (rt2 nhip) dip\}} \) by (rule sqn [symmetric])
  also have \( \text{\{sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))\}} \)
    unfolding nsqn_def using \( \text{\{dip \in vD (rt2 nhip)\}} \)
    by (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show \( \text{\{nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip, \{)) dip)) = nsqn, (the (rt2 nhip dip))\}} \).
next
have "(dhops (rt2 nhip) dip) ≤ hops" by (rule hop)
also have "hops < hops + 1" by simp
also have "hops + 1 = (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: proj_eq_dhops)
qed

lemma nsqn_invalidate:
assumes "dip ∈ kD(rt)"
and "∀ ip ∈ dom(dests). ip ∈ vD(rt) ∧ the (dests ip) = inc (sqn rt ip)"
sows "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

2.6 Invariant proofs on individual processes

theory B_Seq_Invariants
imports "../../../AWN/Invariants" B_Aodv B_Aodv_Data B_Aodv_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:
"paodv i|=A onll Γ_AODV (λ((ξ, _), (_, (ξ', _))). sn ξ ≤ sn ξ')" by inv_cterms

lemma sequence_number_one_or_bigger:
"paodv i|= onl Γ_AODV (λ(ξ, _). 1 ≤ sn ξ)" by (rule onll_step_to_invariantI [OF sequence_number_increases])
(auto simp: σ_AODV_def)

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

lemma sequence_number_increases':
"paodv i|= (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 Γ_AODV (λ(ξ, 1). 1 ∈ (PAodv−:7} ∪ {PAodv−:5} ∪ {PRrep−:0..PRrep−:4} ∪ {PRreq−:0..PRreq−:3} → sip ξ ∈ kD (rt ξ))" by inv_cterms

lemma addpreRT_partly_welldefined:
"paodv i|= onl Γ_AODV (λ(ξ, 1). 1 ∈ {PRreq−:16..PRreq−:18} ∪ {PRrep−:1..PRrep−:5} → dip ξ ∈ kD (rt ξ))"
\( l \in \{ \text{PRreq-:3..PRreq-:17} \} \rightarrow \text{oip } \xi \in kD(\text{rt } \xi)\)"

by \text{inv_cterms}

**Proposition 7.38**

lemma \text{includes_nhip}:

"paodv i \models \text{onl} \Gamma_{\text{AODV}} (\lambda(\xi, l). \forall \text{dip}\in kD(\text{rt } \xi). \text{the } (\text{nhop } (\text{rt } \xi) \text{ dip})\in kD(\text{rt } \xi))"

proof -

{ fix ip and \( \xi \)' : state
  assume "\( \forall \text{dip}\in kD(\text{rt } \xi). \text{the } (\text{nhop } (\text{rt } \xi) \text{ dip})\in kD(\text{rt } \xi)\)"
  and "\( \xi \)' = \( \langle\text{rt} := \text{update } (\text{rt } \xi) \text{ ip} (0, \text{unk}, \text{val}, \text{Suc} 0, \text{ip}, \{\})\rangle\)"
  hence "\( \forall \text{dip}\in kD(\text{rt } \xi). \text{the } (\text{nhop } (\text{update } (\text{rt } \xi) \text{ ip} (0, \text{unk}, \text{val}, \text{Suc} 0, \text{ip}, \{\}) \text{ dip}) = \text{ip} \)
    \lor \text{the } (\text{nhop } (\text{update } (\text{rt } \xi) \text{ ip} (0, \text{unk}, \text{val}, \text{Suc} 0, \text{ip}, \{\}) \text{ dip}) \in kD(\text{rt } \xi)\)"
  by \text{clarsimp} (metis \text{nhop_update_unk_val} \text{update_another})
}

note one_hop = this

{ fix ip sip sn hops and \( \xi \)' : state
  assume "\( \forall \text{dip}\in kD(\text{rt } \xi). \text{the } (\text{nhop } (\text{rt } \xi) \text{ dip})\in kD(\text{rt } \xi)\)"
  and "\( \xi \)' = \( \langle\text{rt} := \text{update } (\text{rt } \xi) \text{ ip} (\text{sn}, \text{kno}, \text{val}, \text{Suc} \text{ hops}, \text{sip}, \{\})\rangle\)"
  and "sip \in kD(\text{rt } \xi)"
  hence "\( \text{the } (\text{nhop } (\text{update } (\text{rt } \xi) \text{ ip} (\text{sn}, \text{kno}, \text{val}, \text{Suc} \text{ hops}, \text{sip}, \{\}) \text{ dip}) = \text{ip} \)
    \lor \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 kD(\text{rt } \xi)\)"
    \land \( \forall \text{dip}\in kD(\text{rt } \xi). \text{the } (\text{nhop } (\text{update } (\text{rt } \xi) \text{ ip} (\text{sn}, \text{kno}, \text{val}, \text{Suc} \text{ hops}, \text{sip}, \{\}) \text{ dip}) = \text{ip} \)
    \lor \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 kD(\text{rt } \xi)\)"
  by (metis \text{kD_update_unchanged} \text{nhop_update_changed} \text{update_another})
}

note nhip_is_sip = this

show \(?\text{thesis}\) by (inv_cterms inv add: \text{onl_invariant_sterms [OF aodv_wf sip_in_kD]}
\text{onl_invariant_sterms [OF aodv_wf addpreRT_partly_welldefined]}
solve: one_hop nhip_is_sip)

qed

**Proposition 7.22**: needed in **Proposition 7.4**

lemma \text{addpreRT_welldefined}:

"paodv i \models \text{onl} \Gamma_{\text{AODV}} (\lambda(\xi, l). (l \in \{ \text{PRreq-:16..PRreq-:18} \} \rightarrow \text{dip } \xi \in kD(\text{rt } \xi))
\land
(l = \text{PRreq-:17} \rightarrow \text{oip } \xi \in kD(\text{rt } \xi))
\land
(l = \text{PRrep-:4} \rightarrow \text{dip } \xi \in kD(\text{rt } \xi))
\land
(l = \text{PRrep-:5} \rightarrow (\text{the } (\text{nhop } (\text{rt } \xi) \text{ dip}) \in kD(\text{rt } \xi)))"

(is \(\_ \models \text{onl} \Gamma_{\text{AODV}} \text{?P}\))

unfolding \text{invariant_def} proof

fix s
then obtain \( \xi p \) where "s = \( \langle\xi, p\rangle\)"
  and "\( \langle\xi, p\rangle \in \text{reachable } (\text{paodv } i) \text{ TT}\)"
  by (metis \text{PairE})

have "\text{onl} \Gamma_{\text{AODV}} \text{?P } (\xi, p)"
  proof (rule \text{onlI})
    fix l
    assume "l \in \text{labels } \Gamma_{\text{AODV}} p"
    with \( \langle\xi, p\rangle \in \text{reachable } (\text{paodv } i) \text{ TT}\)
    have I1: "l \in \{ \text{PRreq-:16..PRreq-:18} \} \rightarrow \text{dip } \xi \in kD(\text{rt } \xi)"
      \land
      (l = \text{PRreq-:17} \rightarrow \text{oip } \xi \in kD(\text{rt } \xi))
      \land
      (l = \text{PRrep-:4} \rightarrow \text{dip } \xi \in kD(\text{rt } \xi))
      \land
      (l = \text{PRrep-:5} \rightarrow (\text{the } (\text{nhop } (\text{rt } \xi) \text{ dip}) \in kD(\text{rt } \xi)))"
    by (auto dest!: \text{invariantD [OF \text{addpreRT_partly_welldefined]})
    moreover from \( \langle\xi, p\rangle \in \text{reachable } (\text{paodv } i) \text{ TT}\) \( l \in \text{labels } \Gamma_{\text{AODV}} p\) and I3
    have "l = \text{PRrep-:5} \rightarrow (\text{the } (\text{nhop } (\text{rt } \xi) \text{ dip}) \in kD(\text{rt } \xi))"
      by (auto dest!: \text{invariantD [OF \text{includes_nhip}]})
    ultimately show "\text{?P } (\xi, l)"
      by \text{simp}
  qed
with \( s = \langle\xi, p\rangle\) show "\text{onl} \Gamma_{\text{AODV}} \text{?P } s"
  by \text{simp}
qed
Proposition 7.4

lemma known_destinations_increase:
"paodv i |=A onll Γ_AODV (λ((ξ, _), _, (ξ', _)). kD (rt ξ) ⊆ kD (rt ξ'))"
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 |=A onll Γ_AODV (λ((ξ, _), _, (ξ', _)). rreqs ξ ⊆ rreqs ξ')"
by (inv_cterms simp add: subset_insertI)

lemma dests_bigger_than_sqn:
"paodv i |= onll Γ_AODV (λ(λ(ξ, l). l ∈ {PAodv-:15..PAodv-:19} ∪ {PPkt-:7..PPkt-:11} ∪ {PRreq-:9..PRreq-:13} ∪ {PRreq-:21..PRreq-:25} ∪ {PRrep-:9..PRrep-:13} ∪ {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:
"paodv i |= onll Γ_AODV (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_inval = 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 |= onll Γ_AODV (λ(ξ, _). ip ξ = i)"
by (inv_cterms simp add: σ_AODV_def)

Proposition 7.8

lemma sender_ip_valid":
lemma sender_ip_valid:
"paodv i |\ A onll \GAMMA_{AODV} (\lambda((\xi, \_), a, \_). anycast (\lambda m. not_Pkt m \rightarrow msg_sender m = ip (\xi) a))" by inv_cterms

Proposition 7.9

lemma sip_not_ip':
"paodv i |\ A onll \GAMMA_{AODV} (\lambda((\xi, \_), a, \_). anycast (\lambda m. not_Pkt m \rightarrow 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:
"paodv i |\ (recvmsg P \rightarrow) onl \GAMMA_{AODV} (\lambda(\xi, 1). l \in \{PAodv-:1\} \rightarrow P (msg (\xi)))"
by inv_cterms

Proposition 7.10

lemma hop_count_positive:
"paodv i |\ A onll \GAMMA_{AODV} (\lambda((\xi, \_), a, \_). anycast (\lambda m. not_Pkt m \rightarrow msg_sender m = ip \xi a))"
by inv_cterms

Neither sip_not_ip' nor sip_not_ip is needed to show loop freedom.

proof (inv_cterms, elim conjE)
fix \_ l \xi pp p'
assume "((\xi, pp) \in reachable (paodv i) TT" 
and "\{PRreq-:17\}[\lambda \xi. \xi\{rt := \text{the (addpreRT (rt \xi) (oip \xi) \{\text{the (nhop (rt \xi) (dip \xi)))\})}\}] p' \in st" 
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 aodv_wf addpreRT_welldefined], l="PRreq-:17")
with "dip \xi \in vD (rt \xi)"
show "dip \xi \in vD (the (addpreRT (rt \xi) (oip \xi) \{\text{the (nhop (rt \xi) (dip \xi)))\}))" by simp

qed

lemma rreq_dip_in_vD_dip_eq_ip:
"paodv i |\ onll \GAMMA_{AODV} (\lambda(\xi, 1). 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 (paodv i) TT" 
and "\{PRreq-:17\}[\lambda \xi. \xi\{rt := \text{the (addpreRT (rt \xi) (oip \xi) \{\text{the (nhop (rt \xi) (dip \xi)))\})}\}] p' \in st" 
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 aodv_wf addpreRT_welldefined], l="PRreq-:17")
with "dip \xi \in vD (rt \xi)"
show "dip \xi \in vD (the (addpreRT (rt \xi) (oip \xi) \{\text{the (nhop (rt \xi) (dip \xi)))\}))" by simp

lemma rrep_dip_in_vD:
"paodv i |\ onll \GAMMA_{AODV} (\lambda(\xi, 1). l \in \{PRrep-:4..PRrep-:6\} \rightarrow dip \xi \in vD (rt \xi)))"

proof (inv_cterms)
fix \_ l \xi pp p'
assume "((\xi, pp) \in reachable (paodv i) TT" 
and "\{PRreq-:17\}[\lambda \xi. \xi\{rt := \text{the (addpreRT (rt \xi) (oip \xi) \{\text{the (nhop (rt \xi) (dip \xi)))\})}\}] p' \in st" 
and "l = PRreq-:17"
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 aodv_wf addpreRT_welldefined], l="PRreq-:17")
with "dip \xi \in vD (rt \xi)"
show "dip \xi \in vD (the (addpreRT (rt \xi) (the (nhop (rt \xi) (dip \xi))) \{\text{the (nhop (rt \xi) (oip \xi)))\}))" by simp
Proposition 7.11

lemma anycast_msg_zhops:

\[ \text{\texttt{\textbackslash \textbackslash} reqid dip dsn dsk oip osn sip.} \]
\[ \text{paodv i \texttt{\textbackslash \textbackslash} onll} \Gamma_{AODV} (\lambda(l, a, \_). \text{anycast msg_zhops a}) \]

proof

\( \text{\texttt{\textbackslash (invcterns inv add:}} \)
\[ \text{onl_invariant_sterms [OF aodv wf rreq dip in vD dip eq ip]} \]
\[ \text{onl_invariant_sterms [OF aodv wf rrep dip in vD]} \]
\[ \text{onl_invariant_sterms [OF aodv wf hop count positive],} \]
\[ \text{elim conjE} \]
\[ \text{fix} \ l \ \xi \ \text{a pp} \ p' \ \text{pp'} \]
\[ \text{assume} \ \text{"(ξ, pp) \in reachable (paodv i) TT"} \]
\[ \text{and} \ \text{"(PRreq-:18)unicast(λξ. the (nhop (rt ξ) (oip ξ)),} \]
\[ \text{λξ. Rrep (the (dhops (rt ξ) (dip ξ))) (dip ξ) (sqn (rt ξ) (dip ξ)) (oip ξ) (ip ξ)).} \]
\[ p' > pp' \in \text{sterms} \Gamma_{AODV} \text{pp} \]
\[ \text{and} \ \text{"1 = PRreq-:18"} \]
\[ \text{and} \ \text{"a = unicast (the (nhop (rt ξ) (oip ξ))),} \]
\[ \text{(Rrep (the (dhops (rt ξ) (dip ξ))) (dip ξ) (sqn (rt ξ) (dip ξ)) (oip ξ) (ip ξ))"} \]
\[ \text{and} \ \text{"∀ip \in kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) (ip ξ))"} \]
\[ \text{and} \ \text{"dip ξ \in vD (rt ξ)"} \]
\[ \text{from} \ \text{dip ξ ∈ vD (rt ξ)} \text{have} \ \text{"dip ξ ∈ kD (rt ξ)"} \]
\[ \text{by (rule vD id gives kD(1))} \]
\[ \text{with} \ \text{* have} \ \text{"Suc 0 ≤ the (dhops (rt ξ) (dip ξ))"} \]
\[ \text{thus} \ \text{"0 < the (dhops (rt ξ) (dip ξ))"} \text{by simp} \]
\[ \text{next} \]
\[ \text{fix} \ l \ \xi \ \text{a pp} \ p' \ \text{pp'} \]
\[ \text{assume} \ \text{"(ξ, pp) \in reachable (paodv i) TT"} \]
\[ \text{and} \ \text{"(PRrep-:6)unicast(λξ. the (nhop (rt ξ) (oip ξ)),} \]
\[ \text{λξ. Rrep (the (dhops (rt ξ) (dip ξ))) (dip ξ) (sqn (rt ξ) (dip ξ)) (oip ξ) (ip ξ)).} \]
\[ p' > pp' \in \text{sterms} \Gamma_{AODV} \text{pp} \]
\[ \text{and} \ \text{"1 = PRrep-:6"} \]
\[ \text{and} \ \text{"a = unicast (the (nhop (rt ξ) (oip ξ))),} \]
\[ \text{(Rrep (the (dhops (rt ξ) (dip ξ))) (dip ξ) (sqn (rt ξ) (dip ξ)) (oip ξ) (ip ξ))"} \]
\[ \text{and} \ \text{"∀ip \in kD (rt ξ). Suc 0 ≤ the (dhops (rt ξ) (ip ξ))"} \]
\[ \text{and} \ \text{"dip ξ \in vD (rt ξ)"} \]
\[ \text{from} \ \text{dip ξ ∈ vD (rt ξ)} \text{have} \ \text{"dip ξ ∈ kD (rt ξ)"} \]
\[ \text{by (rule vD id gives kD(1))} \]
\[ \text{with} \ \text{* have} \ \text{"Suc 0 ≤ the (dhops (rt ξ) (dip ξ))"} \]
\[ \text{thus} \ \text{"the (dhops (rt ξ) (dip ξ)) = 0 \rightarrow dip ξ = ip ξ"} \text{by simp} \]
\[ \text{by auto} \]

qed

lemma hop_count_zero_oip_dip_sip:

\[ \text{\texttt{\textbackslash (paodv i \texttt{\textbackslash (recmsg msg_zhops \rightarrow) onll} \Gamma_{AODV} (λ(ξ, 1).} \]
\[ \text{(l∈\{AODV-:4,AODV-:5\}) \cup \{PRreq-:n\|n. True\} \rightarrow} \]
\[ \text{(dhops ξ = 0 \rightarrow oip ξ = sip ξ))} \]
\[ \text{∧} \]
\[ \text{(l∈\{AODV-:6,AODV-:7\}) \cup \{PRrep-:n\|n. True\} \rightarrow} \]
\[ \text{(dhops ξ = 0 \rightarrow dip ξ = sip ξ))"} \]
\[ \text{by (invcterns inv add: onll_invariant_sterms [OF aodv wf received_msg_inv]) auto} \]

lemma osn_rreq:

\[ \text{\texttt{\textbackslash (paodv i \texttt{\textbackslash (recmsg rreq rrep sn \rightarrow) onll} \Gamma_{AODV} (λ(ξ, 1).} \]
\[ \text{l ∈ \{AODV-:4, AODV-:5\} \cup \{PRreq-:n\|n. True\} \rightarrow 1 ≤ osn ξ)"} \]
\[ \text{by (invcterns inv add: onll_invariant_sterms [OF aodv wf received_msg_inv]) clarsimp} \]

lemma osn_rreq':

\[ \text{\texttt{\textbackslash (paodv i \texttt{\textbackslash (recmsg (λm. rreq rrep sn m \& msg_zhops m) \rightarrow) onll} \Gamma_{AODV} (λ(ξ, 1).} \]
\[ \text{l ∈ \{AODV-:4, AODV-:5\} \cup \{PRreq-:n\|n. True\} \rightarrow 1 ≤ osn ξ)"} \]
\[ \text{proof (rule invariant weakenE [OF osn rreq])} \]
\[ \text{fix} \ a \]
\[ \text{assume} \ \text{"recmsg (λm. rreq rrep sn m \& msg_zhops m) a"} \]
\[ \text{thus} \ \text{"recmsg rreq rrep sn a"} \]
by (cases a) simp_all
qed

lemma dsn_rrep:
"paodv i |= (recvmsg rreq_rrep_sn ⟷ onl Γ_AODV (λ(ξ, l).
1 ∈ {PAodvː6, PAodvː7} ∪ {PRrepːn|n. True} → 1 ≤ dsn ξ))"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv])clarsimp

lemma dsn_rrep':
"paodv i |= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) → onl Γ AODV (λ(ξ, l).
l ∈ {PAodvː4..PAodvː5} ∪ {PRreqːn|n. True} −→ (hops ξ = 0 −→ oip ξ = sip ξ)) ∧
((l ∈ {PAodvː6..PAodvː7} ∪ {PRrepːn|n. True} →
(hops ξ = 0 −→ dip ξ = sip ξ))))"
proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])
fix a
assume "recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a"
thus "recvmsg msg_zhops a"
by (cases a) simp_all
qed

Proposition 7.12
lemma zero_seq_unk_hops_one':
"paodv i |= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) → onl Γ_AODV (λ(ξ, _).
∀ dip ∈ kD(rt ξ). (sqn (rt ξ) dip = 0 −→ sqnf (rt ξ) dip = unk)
∧ (sqnf (rt ξ) dip = unk −→ the (dhops (rt ξ) dip) = 1)
∧ (the (dhops (rt ξ) dip) = 1 −→ the (nhop (rt ξ) dip) = dip)))"
proof -
{ fix dip and ξ :: 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
\begin{verbatim}
{ 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, simp_all add: sqn_def sqnf_def)
} 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)) = Suc 0))" 
 by - (rule update_cases) simp_all
} 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_TT [OF aodv_wf addpreRT_welldefined]
 onl_invariant_sterms [OF aodv_wf hop_count_zero_oip_dip_sip']
 seq_step_invariant_sterms_TT [OF sqns_increase aodv_wf aodv_trans]
 onl_invariant_sterms [OF aodv_wf osn_rreq']
 onl_invariant_sterms [OF aodv_wf dsn_rrep']) clarsimp*
qed

lemma zero_seq_unk_hops_one:
 "paodv i || (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) → onl Γ_{AODV} (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

215
\end{verbatim}
lemma kD_unk_or_atleast_one:  
"paoav i |= (recvmsg rreq_rrep_sn →) onl Γ_{AODV} (λ(ξ, 1).  
\forall dip∈kD(rt ξ). π_3(the (rt ξ dip)) = unk ∨ 1 ≤ π_2(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 "π_3(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∈kD(rt). π_3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip"  
  and **: "dsk1 = unk ∨ Suc 0 ≤ dsn2"  
  have "∀ dip∈kD(rt). π_3(the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1, pre1) dip)) = 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 ?thesis  
  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). π_3(the (rt ip)) = unk ∨ Suc 0 ≤ sqn rt ip"  
  have "∀ dip∈kD(rt). π_3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests dip)"  
  proof  
    fix dip  
    assume "dip∈kD(rt)"  
    with ** have "π_3(the (rt dip)) = unk ∨ Suc 0 ≤ sqn rt dip" ..  
    thus "π_3 (the (rt dip)) = unk ∨ Suc 0 ≤ sqn (invalidate rt dests dip)"  
  proof  
    assume "π_3(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 \|\|_A (recvmsg rreq_rrep_sn \rightarrow) onll \Gamma_{AODV} (\lambda(_, a, _). anycast rreq_rrep_sn a)"
proof (clarsimp)
have sqnf_kno: "paodv i \|\|_A onll \Gamma_{AODV} (\lambda(\xi, \ell). anycast (rreq_rrep_sn \xi) a)"
by (inv_cterms inv add: onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined])
have rrep_sqn_greater_dsn: "paodv i \|\|_A (recvmsg rreq_rrep_sn \rightarrow) onll \Gamma_{AODV} (\lambda(\xi, \ell). anycast (rreq_rrep_sn \xi) a)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv] onl_invariant_sterms_TT [OF aodv_wf addpreRT_welldefined] onl_invariant_sterms [OF aodv_wf dsn_rrep])

by (clarsimp simp: update_kno_dsn_greater_zero [simplified])

show ?thesis

(auto simp: proj2_eq_sqn)

qed

Proposition 7.14

lemma rreq_rrep_fresh_any_step_invariant:
"paodv i \|\|_A onll \Gamma_{AODV} (\lambda((\xi, \ell), a, _). anycast (rreq_rrep_fresh (rt \xi)) a)"
proof

have rreq_oip: "paodv i \|\|_A onll \Gamma_{AODV} (\lambda(\xi, \ell). anycast (rreq_rrep_fresh (rt \xi)) a)"
proof (clarsimp)
fix 1 \xi 1' pp p'
assume "((\xi, pp) \in reachable (paodv i)) TT" and "(PRreq:-2)\xi.[\xi rt := update (rt \xi) (oip \xi) (osn \xi, kno, val, Suc (hops \xi), sip \xi, {})] p' \in stterms \Gamma_{AODV} pp" and "1' = 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)

(auto simp: proj2_eq_sqn)

qed

have rrep_prrep: "paodv i \|\|_A onll \Gamma_{AODV} (\lambda(\xi, 1).

(1 \in \{PRreq:-4, PRreq:-6\} \rightarrow (dip \xi \in kD(rt \xi) \wedge the (flag (rt \xi) (dip \xi)) = val)))"
by (inv_cterms inv add: onl_invariant_sterms TT [OF aodv_wf addpreRT_welldefined] onl_invariant_sterms [OF aodv_wf sip_in_kD])


qed

Proposition 7.15

lemma rerr_invalid_any_step_invariant:
"paodv i =⇒ onl Γ AODV (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a)"
proof -
  have dests_inv: "paodv i =⇒ onl Γ AODV (λ((ξ, l), a, _). anycast (rerr_invalid (rt ξ)) a)
      (∀ ip∈dom(dests ξ). ip∈vD(rt ξ)))
    ∧ (l ∈ {PAodv:-16..PAodv:-19}
        ∪ {PPkt:-8..PPkt:-11}
        ∪ {PReq:-10..PReq:-13}
        ∪ {PReq:-22..PReq:-25}
        ∪ {PRep:-10..PRep:-13}
        ∪ {PErr:-2..PErr:-5} → (∀ ip∈dom(dests ξ). ip∈iD(rt ξ)
      ∧ the (dests ξ ip) = sqn (rt ξ ip)))
  by inv_cterms (clarsimp split: split_if_asm option.split_asm simp: domIff)+

show ?thesis by (inv_cterms inv add: onl_invariant_sterms [OF aodv_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 ↦ 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:
"paodv i =⇒ onl Γ AODV (λ(ξ, l). (1 ∈ {PAodv:-15, PPkt:-7, PReq:-9, PReq:-21, PRep:-9, PErr:-1}
      → (∀ ip∈dom(dests ξ). ip∈vD(rt ξ))
    ∧ (l = PRreq:-2 → (∀ ip∈dom(dests ξ). ip∈iD(rt ξ)
      ∧ the (dests ξ ip) > sqn (rt ξ ip)))))
  by inv_cterms (clarsimp split: split_if_asm option.split_asm simp: domIff)+

Proposition 7.27

lemma route_tables_fresher:
"paodv i =⇒ (recvmsg rreq_rrep_sn → onl Γ AODV (λ((ξ, _, _), (ξ', _)). ∀ dip∈kD(rt ξ). rt ξ ⊆ dip rt ξ'))"
proof (inv_cterms inv add:
  onl_invariant_sterms [OF aodv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]]
  onl_invariant_sterms [OF aodv_wf osn_rreq]
  onl_invariant_sterms [OF aodv_wf dsn_rreq]
  onl_invariant_sterms [OF aodv_wf addpreRT_welldefined [THEN invariant_restrict_inD]])

fix ξ pp p' assume "((ξ, pp) ∈ reachable (paodv i) (recvmsg rreq_rrep_sn))" and "{PReq:-2}[(ξ, rt := update (rt ξ) (oip ξ) (osn ξ, kno, val, Suc (hops ξ), sip ξ, {})))"
p' ∈ \text{sterms} \Gamma_{AODV} 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 "∀ip∈kD (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 (paodv i) (recvmsg rreq rrep sn)

and "∀ip∈kD (rt ξ). Suc 0 ≤ dsn ξ"

and ∀ip∈kD (rt ξ). rt ξ ⊑ ip update (rt ξ) (oip 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 "∀ip∈kD (rt ξ) (dip ξ) (dsn ξ, kno, val, Suc (hops ξ), sip ξ, { })"

by (rule rt_fresher_update)

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) \( \sqsubseteq_{dip} \) 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 \( \xi' \)) \( \land \) 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 \( \vdash \) ⟨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 \( \xi' \)) \( \land \) 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"

220
shows \[s < nsqn \(rt \xi\) i]\n
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 \lor (s = nsqn \(rt \xi'\) i \land 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\(rt \sigma sip\) \land sn \leq nsqn \(rt \sigma sip\) ip \land (nsqn \(rt \sigma sip\) ip = sn \rightarrow (the \(dhops \(rt \sigma sip\) ip\) \leq hops \lor the \(flag \(rt \sigma sip\) ip\) = inv))"
shows "ip \(\in\) kD\(rt \sigma' sip\) \land sn \leq nsqn \(rt \sigma' sip\) ip \land (nsqn \(rt \sigma' sip\) ip = sn \rightarrow (the \(dhops \(rt \sigma' sip\) ip\) \leq hops \lor the \(flag \(rt \sigma' sip\) ip\) = inv))"
(is "\(_ \land ?nsqnafter\)"
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 \(ip \(\in\) kD\(rt \sigma sip\)) and quality_increases \(\sigma sip\) \(\sigma' sip\)
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 \(ip \(\in\) kD\(rt \sigma sip\)) and quality_increases \(\sigma sip\) \(\sigma' sip\)
have "sn < nsqn \(rt \sigma' sip\) ip \lor (sn = nsqn \(rt \sigma' sip\) ip \land the \(dhops \(rt \sigma' sip\) ip\) \leq the \(dhops \(rt \sigma sip\) ip\) )" ..
hence "sn < nsqn \(rt \sigma' sip\) ip \lor (nsqn \(rt \sigma' sip\) ip = sn \land the \(dhops \(rt \sigma' sip\) ip\) \leq hops \lor 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 \land the \(dhops \(rt \sigma sip\) ip\) \geq the \(dhops \(rt \sigma' sip\) ip\)"
hence "sn = nsqn \(rt \sigma' sip\) 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 \[\lor the (flag (rt (σ sip)) ip) = inv\]"

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

with ⟨ip∈kD(rt (σ' sip))⟩ show ?thesis

proof (rule vD_or_iD)
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 ⟨nsqn (rt (σ sip)) ip > 1⟩
    have "nsqn (rt (σ' sip)) ip > sqn (rt (σ sip)) ip" 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

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 "\(\forall j. quality_increases (σ j) (σ' j)\)"
and "1 ≤ sn"
and *: "ip∈kD(σ sip) \land sn ≤ nsqn (σ sip) ip
\land (nsqn (σ sip) ip = sn
\rightarrow \text{(the (dhops (σ sip) ip) ≤ hops \lor the (flag (σ sip) ip) = inv)})"
shows "ip∈kD(σ' sip) \land sn ≤ nsqn (σ' sip) ip
\land (nsqn (σ sip) ip = sn
\rightarrow \text{(the (dhops (σ' sip) ip) ≤ hops \lor the (flag (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:

222
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 rreqid dip dsn dsk oip osn sip ⇒
oipc ∈ KD(rt (σ sip)) ∧ nsqn (rt (σ sip)) oipc ≥ osnc ∧
(osnc ≥ 1 ∧ (sip ≠ oip → oip ∈ KD(rt (σ sip)))
→ (hops ≥ the (dhops (σ sip)) oip)
∨ the (flag (rt (σ sip)) oip) = inv)))
| Rrep hops dip dsn dsk oip osn sip ⇒
dipc ∈ KD(rt (σ sip)) ∧ nsqn (rt (σ sip)) dipc ≥ dsn ∧
(dsnc ≥ 1 ∧ (sip ≠ dipc → dipc ∈ KD(rt (σ sip)))
→ (dipc ≥ the (dhops (σ sip)) dipc)
∨ the (flag (rt (σ sip)) dipc) = inv)))
| Rerr destsc sipc ⇒
(∀ ripc ∈ dom(destsc). ripc ∈ KD(rt (σ sip)) ∧
nsqn (rt (σ sip)) ripc ≥ osnc ∧
(nsqn (rt (σ sip)) ripc = osnc → (hops ≥ the (dhops (rt (σ sip)) ripc)
∨ the (flag (rt (σ sip)) ripc) = inv)))
| _ ⇒ True"

lemma msg_fresh [simp]:
"∀ hops rreqid dip dsk oip osn sip. msg_fresh σ (Rreq hops rreqid dip dsk oip osn sip) =
(osnc ≥ 1 ∧ nipc ∈ KD(rt (σ sip)) ∧ nsqn (rt (σ sip)) oipc ≥ osnc ∧
(nsqn (rt (σ sip)) oipc = osnc
→ (hops ≥ the (dhops (σ sip)) oip)
∨ the (flag (rt (σ sip)) oip) = inv)))"
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

todo
by clarsimp
have \((\forall \text{rip}\in\text{dom(dests)}. (\text{rip}\in\text{dom(dests)})\land \text{the (dests rip) - 1 } \leq \text{nsqn (rt (\sigma sip)) rip}))\)

proof
fix rip
assume "rip \in \text{dom dests}"
with * have "rip\in\text{iD(rt (\sigma sip))} and \text{the (dests rip) = sqn (rt (\sigma sip)) rip}"
from this(2) have "the (dests rip) - 1 = sqn (rt (\sigma sip)) rip" by simp

finally have "the (dests rip) - 1 \leq \text{nsqn (rt (\sigma sip)) rip}"

with \(\text{rip}\in\text{iD(rt (\sigma sip))}\)
show "rip\in\text{kd(rt (\sigma sip))} \land \text{the (dests rip) - 1 } \leq \text{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 = \text{Rreq hops rreqid dip dsn dsk oip osn sip}"
and "msg_fresh \sigma m"
then have "\text{osn} \geq 1" and "\text{osn} \leq \text{nsqn (rt (\sigma sip)) oip}
\land (\text{nsqn (rt (\sigma sip)) oip} = \text{osn}
\rightarrow (\text{the (dhops (rt (\sigma sip)) oip)} \leq \text{hops}
\lor \text{the (flag (rt (\sigma sip)) oip) = inv}))"

by auto

from this(2) show ?thesis

proof
assume "sip = oip" with \(\text{osn} \geq 1\) show ?thesis by simp
next
assume "oip\in\text{kd(rt (\sigma sip))} \land \text{osn} \leq \text{nsqn (rt (\sigma sip)) oip}
\land (\text{nsqn (rt (\sigma sip)) oip} = \text{osn}
\rightarrow (\text{the (dhops (rt (\sigma sip)) oip)} \leq \text{hops}
\lor \text{the (flag (rt (\sigma sip)) oip) = inv}))"

moreover from qinc have "quality_increases (\sigma sip) (\sigma' sip)" ..
ultimately have "oip\in\text{kd(rt (\sigma sip))} \land \text{osn} \leq \text{nsqn (rt (\sigma sip)) oip}
\land (\text{nsqn (rt (\sigma sip)) oip} = \text{osn}
\rightarrow (\text{the (dhops (rt (\sigma sip)) oip)} \leq \text{hops}
\lor \text{the (flag (rt (\sigma sip)) oip) = inv}))"

using \(\text{osn} \geq 1\) by (rule quality_increases_rreq_rrep_props [rotated 2])
with \(\text{osn} \geq 1\) show "msg_fresh \sigma' m"
by (clarsimp)

qed
next
fix hops dip dsn oip sip
assume [simp]: "m = \text{Rrep hops dip dsn oip sip}"
and "msg_fresh \sigma m"
then have "\text{dsn} \geq 1" and "\text{dsn} \leq \text{nsqn (rt (\sigma sip)) dip}
\land (\text{nsqn (rt (\sigma sip)) dip} = \text{dsn}
\rightarrow (\text{the (dhops (rt (\sigma sip)) dip)} \leq \text{hops}
\lor \text{the (flag (rt (\sigma sip)) dip) = inv}))"

by auto
from this(2) show ?thesis

proof
assume "sip = dip" with \(\text{dsn} \geq 1\) 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))"
mOREVER 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
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_Lables" "./././AWN/OAWN_Convert"
begin
Definitions for stating and proving global network properties over individual processes.
definition σ_AODV' :: "((ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp)) set"
where "σ_AODV' ≡ {λi. aodv_init i, Γ_AODV PAodv}"
abbreviation opaodv :: "ip ⇒ ((ip ⇒ state) × (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
where "opaodv i ≡ (|init = σ_AODV', trans = oseqp_sos Γ_AODV i |)"
lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))"
unfolding σ_AODV_def σ_AODV'_def by rule simp_all
lemma oadv_control_within [simp]: "control_within Γ_AODV (init (opaodv i))"
unfolding σ_AODV'_def by (rule control_withinI) (auto simp del: Γ_AODV_simps)
lemma σ_AODV'_labels [simp]: "(σ, p) ∈ σ_AODV' ⇒ labels Γ_AODV p = {PAodv−:0}"
unfolding σ_AODV'_def by simp
lemma oadv_init_kD_empty [simp]: "(σ, p) ∈ σ_AODV' ⇒ kD (rt (σ i)) = {}"
unfolding σ_AODV'_def kD_def by simp
global invariant proofs over sequential processes

theory B_Global_Invariants
imports B_Seq_Invariants
B_Aodv_Predicates
B_Fresher
B_Quality_Increases
"../../../AWN/OAWN_Convert"
B_OAodv
begin

lemma other_quality_increases [elim]:
  assumes "other quality_increases I σ σ'"
  shows "∀ j. quality_increases (σ j) (σ' j)"
  using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
  fixes m
  assumes *: "otherwith P I (orecvmsg Q) σ σ' a" and
  weakenP: "∀ σ m. P σ m =⇒ P' σ m" and
  weakenQ: "∀ σ m. Q σ m =⇒ Q' σ m"
  shows "otherwith P' I (orecvmsg Q') σ σ' a"
  proof
  fix j
  assume "j /∈ I" with * have "P (σ j) (σ' j)" by auto
  thus "P' (σ j) (σ' j)" by (rule weakenP)
  qed

lemma oreceived_msg_inv:
  assumes other: "∀ σ σ' m. [ [ P σ m; other Q {i} σ σ' ] =⇒ P σ' m ]" and
  local: "∀ σ m. P σ m =⇒ P (σ (i := σ i (msg := m))) m"
  shows "opaodv i = (otherwith Q {i} (orecvmsg P), other Q {i} →)
  onl Γ_AODV (λ(σ, l). l = PAodv-:1 −→ P σ (msg (σ i))"
  proof (inv_cterms, intro impI)
    fix σ σ' l
    assume "l = PAodv-:1 =⇒ P σ (msg (σ i))" and
    "l = PAodv-:1" and
    "other Q {i} σ σ'"
    from this(1-2) have "P σ (msg (σ i))" ...
    moreover have "σ' i = σ i" using other Q {i} σ σ' by (rule other)
    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 \(\sigma' i = \sigma i | (\text{msg} := \text{msg})\)

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 \(P (\sigma(i := \sigma i | (\text{msg} := \text{msg}))) \text{msg}\)

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:

"paodv i \|\|= 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 \text{reachable} (paodv i) (recvmsg rreq_rrep_sn)"

and tr: "(s, a, s') \in \text{trans} (paodv i)"

and rm: "recvmsg rreq_rrep_sn a"  

from sr have srTT: "s \in \text{reachable} (paodv i) TT" ..

from route_tables_fresher sr tr rm  

have "onll \(\Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{dip} \in \text{kD} (rt \xi). \text{rt} \xi \sqsubseteq \text{dip} \text{rt} \xi') (s, a, s')\)"

by (rule step_invariantD)

moreover from known_destinations_increase srTT tr TT_True  

have "onll \(\Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{kD} (rt \xi) \subseteq \text{kD} (rt \xi')) (s, a, s')\)"

by (rule step_invariantD)

moreover from sqns_increase srTT tr TT_True  

have "onll \(\Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{ip}. \text{sqn} (rt \xi) ip \leq \text{sqn} (rt \xi') ip) (s, a, s')\)"

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_aodv oaodv_trans aodv_trans, simplified seqll_onll_swap]

lemma quality_increases:

"opaodv i |=A (otherwith quality_increases {i} (orecvmsg (\_ rreq_rrep_sn))),  
other_quality_increases {i} \rightarrow \(\text{onll} \Gamma_{AODV} (\lambda((\sigma, _), _, (\sigma', _)). \forall j. \text{quality_increases} (\sigma j) (\sigma' j))\)"

(is "\_|=A (?S, _ \rightarrow _)"

proof (rule onll_ostep_invariantI, simp)

fix \(\sigma \ p \ a \ \sigma' \ p' \ l\)

assume or: "(\sigma, p) \in \text{oreachable} (opaodv i) ?S \ (other \text{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 (\_ rreq_rrep_sn) \sigma a"

by (auto dest!: ooreachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"

and QU="other \text{quality_increases} \{i\}]"

otherwith_actionD)

with or have orw: "(\sigma, p) \in \text{oreachable} (opaodv i) \ (act (recvmsg rreq_rrep_sn))\"

(other \text{quality_increases} \{i\})"

by - (erule ooreachable_weakenE, auto)

with tr ll ll' and orecvmsg (\_ 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:
"opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} →)
onll Γ_AODV (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 (opaodv i) (act (recvmsg rreq_rrep_sn)) (other A {i}))" and "((σ, p), a, (σ', p')) ∈ oseqp_sos Γ_AODV i"
  and recv: "act (recvmsg rreq_rrep_sn) σ σ' a"
  obtain l l' where "l ∈ labels Γ_AODV p" and "l' ∈ labels Γ_AODV p'" by (metis aodv_ex_label)
  from ⟨((σ, p), a, (σ', p')) ∈ oseqp_sos Γ_AODV i⟩ have tr: "((σ, p), a, (σ', p')) ∈ trans (opaodv i)" by simp
  have "anycast (rreq_rrep_fresh (rt (σ i))) a"
    proof
      have "opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} →) onll Γ_AODV (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_aodv, simplified seqll_onll_swap]]) auto
      hence "onll Γ_AODV (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 Γ_AODV p⟩ and ⟨l' ∈ labels Γ_AODV p'⟩ by auto
    qed

moreover have "anycast (rerr_invalid (rt (σ i))) a"
  proof -
    have "opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} →) onll Γ_AODV (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_aodv, simplified seqll_onll_swap]]) auto
    hence "onll Γ_AODV (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 Γ_AODV p⟩ and ⟨l' ∈ labels Γ_AODV p'⟩ by auto
  qed

moreover have "anycast rreq_rrep_sn a"
  proof -
    from or tr recv have "onll Γ_AODV (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_aodv, simplified seqll_onll_swap], oadv_trans aodv_trans, simplified seqll_onll_swap])
    thus ?thesis
      using ⟨l ∈ labels Γ_AODV p⟩ and ⟨l' ∈ labels Γ_AODV p'⟩ by auto
  qed

moreover have "anycast (λm. not_Pkt m → msg_sender m = i) a"
  proof -
    have "opaodv i \models_A (act (recvmsg rreq_rrep_sn), other A \{i\} →) onll Γ_AODV (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 Γ_AODV p⟩ and ⟨l' ∈ labels Γ_AODV p'⟩ by auto
  qed

229
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 \(\Gamma_{AODV}(\lambda((\sigma, _), a, _). \text{anycast (msg\_fresh} \sigma) a)((\sigma, p), a, (\sigma', p'))\)"
by auto
qed

lemma oreceived_rreq_rrep_nsqn_fresh_inv:
"opaodv i \[=\] \(\text{otherwith quality\_increases \{i\} (orecvmsg msg\_fresh), other quality\_increases \{i\} \rightarrow onll }\) \(\Gamma_{AODV}(\lambda((\sigma, _), a, _). \text{anycast (msg\_fresh} \sigma) a)((\sigma, p), a, (\sigma', p'))\)"
proof (rule oreceived_msg_inv)
fix \(\sigma \sigma' m\)
assume \(*: \text{msg\_fresh }\sigma m\)
and \("\text{other quality\_increases \{i\} }\sigma \sigma'\"
from this(2) have "\(\forall j. \text{quality\_increases }\sigma j (\sigma' j)\)"..
thus "\text{msg\_fresh }\sigma' m\) using *..
next
fix \(\sigma m\)
assume "\text{msg\_fresh }\sigma m"
thus "\text{msg\_fresh }\sigma (i := \sigma i (|msg := m)) m"
proof (cases m)
fix dests sip
assume "\(m = Rerr \text{dests sip}\)"
with \("\text{msg\_fresh }\sigma m\) show ?thesis by auto
qed auto
qed

lemma oquality_increases_nsqn_fresh:
"opaodv i \[=\] \(\text{otherwith quality\_increases \{i\} (orecvmsg msg\_fresh), other quality\_increases \{i\} \rightarrow onll }\) \(\Gamma_{AODV}(\lambda((\sigma, _), a, _). \forall j. \text{quality\_increases }\sigma j (\sigma' j))\)"
by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
"opaodv i \[=\] \(\text{otherwith quality\_increases \{i\} (orecvmsg msg\_fresh), other quality\_increases \{i\} \rightarrow onll }\) \(\Gamma_{AODV}(\text{seql }i (\lambda(\xi, l). l \in \{PAodv-:4, PAodv-:5\} \cup \{PRreq-:n \mid \text{n. True}\} \rightarrow 1 \leq osn }\))"
by (rule oinvariant_weakenE [OF open_seq_invariant [OF osn_rreq initiali_aodv]])
(auto simp: seql_onl_swap)

lemma rreq_sip:
"opaodv i \[=\] \(\text{otherwith quality\_increases \{i\} (orecvmsg msg\_fresh), other quality\_increases \{i\} \rightarrow onll }\) \(\Gamma_{AODV}(\text{seql }i (\lambda(\xi, l). l \in \{PAodv-:4, PAodv-:5, PRreq-:0, PRreq-:2\} \land \text{sip }\sigma i \neq oip }\ (\sigma i))\)
\(\rightarrow oip (\sigma i) \in kD(rt (\sigma (\text{sip }\sigma i))))\)
\(\land \text{nsqn }\(\text{rt }\sigma (\text{sip }\sigma i))\) (oip (\sigma i)) \geq osn (\sigma i)\)
\(\land \text{nsqn }\(\text{rt }\sigma (\text{sip }\sigma i))\) (oip (\sigma i)) = osn (\sigma i)\)
\(\rightarrow (\text{hops }\sigma i \geq \text{the } (\text{dhops }\text{rt }\sigma (\text{sip }\sigma i)) (oip (\sigma i))\)
\(\lor \text{the } (\text{flag }\text{rt }\sigma (\text{sip }\sigma i)) (oip (\sigma i)) = \text{inv})\)"
(is \("\(=\) \(\{?S, \not?U \rightarrow \}_\)\"
proof (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf oaoadv_trans]
onll_oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
onll_oinvariant_sterms [OF aodv_wf oosn_rreq]
simp add: seqlsimp
simp del: [one_nat_def, rule impI])
fix \(\sigma \sigma' p l\)
assume "\((\sigma, p) \in \text{oreachable (opaodv } i) \not\text{?}?\)"
and "1 \in \text{labels } \Gamma_{\text{AODV}} p"
and pre:

\[(= \text{PAodv-:4} \lor 1 = \text{PAodv-:5} \lor 1 = \text{PRreq-:0} \lor 1 = \text{PRreq-:2}) \land \text{sip} (\sigma i) \neq \text{oip} (\sigma i) \]
\[\rightarrow \text{oip} (\sigma i) \in \text{kD} (\text{rt} (\sigma (\text{sip} (\sigma i)))) \]
\[\land \text{osn} (\sigma i) \leq \text{nsqn} (\text{rt} (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i)) \]
\[\land (\text{nsqn} (\text{rt} (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) = \text{osn} (\sigma i) \]
\[\rightarrow \text{the} (\text{dhops} (\text{rt} (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) \leq \text{hops} (\sigma i) \]
\[\lor \text{the} (\text{flag} (\text{rt} (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) = \text{inv})" \]

and "other_quality_increases \{i\} \sigma \sigma'"
and hyp: "(1=\text{PAodv-:4} \lor 1=\text{PAodv-:5} \lor 1=\text{PRreq-:0} \lor 1=\text{PRreq-:2}) \land \text{sip} (\sigma i) \neq \text{oip} (\sigma i)"
(is "?labels \land \text{sip}(\sigma i) \neq \text{oip}(\sigma i)"
from (4) have "\sigma i = \sigma i" ..
with hyp have hyp': "?labels \land \text{sip}(\sigma i) \neq \text{oip}(\sigma i)" by simp
show "\text{osn} (\sigma i) \in \text{kD} (\text{rt} (\sigma (\text{sip} (\sigma i))))
\land \text{osn} (\sigma i) \leq \text{nsqn} (\text{rt} (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))
\land (\text{nsqn} (\text{rt} (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) = \text{osn} (\sigma i)
\rightarrow \text{the} (\text{dhops} (\text{rt} (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) \leq \text{hops} (\sigma i)
\lor \text{the} (\text{flag} (\text{rt} (\sigma (\text{sip} (\sigma i)))) (\text{oip} (\sigma i))) = \text{inv})"
proof (cases "\text{sip} (\sigma i) = \sigma i")
assume "\text{sip} \neq \sigma i" from (other_quality_increases \{i\} \sigma \sigma’, have "quality_increases \{\text{sip} \sigma i\} \sigma \sigma’"
from hyp show (otherwith quality_increases \{i\} \sigma \sigma’)
by (rule otherE) (clarsimp simp add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf odsn_rrep]
proof (cases "\text{sip} \neq \sigma i")
assume (otherwith quality_increases \{i\} \sigma \sigma’, have "quality_increases \{\text{sip} \sigma i\} \sigma \sigma’"
by (rule otherE) (clarsimp simp add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf odsn_rrep]
proof
fix $\sigma \sigma' p l$
assume "$(\sigma, p) \in \text{reachable } (\text{opao}d i) ?S ?U$
and "$1 \in \text{labels } \Gamma_{AODV} p$"
and pre:

\[(1 = \text{PAoV}v : \sigma) \land (1 = \text{PAoV}d : \sigma) \land (1 = \text{PRrep} : 0) \land (1 = \text{PRrep} : 1) \land \text{zip } (\sigma i) \neq \text{dip } (\sigma i) \]
\[\rightarrow \text{dsn } (\sigma i) \in kD (\text{rt } (\sigma (\text{zip } (\sigma i)))) \]
\[\land \text{dsn } (\sigma i) \leq \text{nsq}n (\text{rt } (\sigma (\text{zip } (\sigma i)))) (\text{dip } (\sigma i)) \]
\[\land (\text{dsn } (\sigma i)) = \text{dsn } (\sigma i) \]
\[\rightarrow \text{the } (\text{dhop } (\text{rt } (\sigma (\text{zip } (\sigma i)))) (\text{dip } (\sigma i))) \leq \text{hop } (\sigma i) \]
\[\lor \text{the } (\text{flag } (\text{rt } (\sigma (\text{zip } (\sigma i)))) (\text{dip } (\sigma i))) = \text{inv}'' \]

and "other quality_increases \{i\} $\sigma \sigma'$"
and hyp: "$(1 = \text{PAoV}v : \sigma) \land (1 = \text{PAoV}d : \sigma) \land (1 = \text{PRrep} : 0) \land (1 = \text{PRrep} : 1) \land \text{zip } (\sigma i) \neq \text{dip } (\sigma i)$"
(is "?labels \land \text{zip } (\sigma i) \neq \text{dip } (\sigma i)"
from (4) have "\(\sigma i = \sigma i" ...
with hyp have hyp': "?labels \land \text{zip } (\sigma i) \neq \text{dip } (\sigma i)" by simp
show "?dip (\sigma i) \in kD (\text{rt } (\sigma (\text{zip } (\sigma i))))" \land \text{dsn } (\sigma i) \leq \text{nsq}n (\text{rt } (\sigma (\text{zip } (\sigma i)))) (\text{dip } (\sigma i)) \land (\text{dsn } (\sigma i)) = \text{dsn } (\sigma i) \rightarrow \text{the } (\text{dhop } (\text{rt } (\sigma (\text{zip } (\sigma i)))) (\text{dip } (\sigma i))) \leq \text{hop } (\sigma i) \lor \text{the } (\text{flag } (\text{rt } (\sigma (\text{zip } (\sigma i)))) (\text{dip } (\sigma i))) = \text{inv}''

proof (cases "\text{zip } (\sigma i) = \text{i}"
assume "\text{zip } (\sigma i) \neq \text{i}"
from "other quality_increases \{i\} $\sigma \sigma'$" have "quality_increases (\sigma (\text{zip } (\sigma i))) (\sigma' (\text{zip } (\sigma i)))"
by (rule otherE) (clarsimp simp: \{\text{zip } (\sigma i) \neq \text{i}\})
moreover from \(\sigma, p) \in \text{reachable } (\text{opao}d i) ?S ?U (\| \in \text{labels } \Gamma_{AODV} p) \land (\text{hyp} \Rightarrow \text{dip } (\sigma i) \neq \text{i})
by (auto dest!: \{\text{onl}_\text{onvariant}_\text{weakenD}[OF \text{dsn}_\text{rrep}]\})
moreover from \(\text{zip } (\sigma i) \neq \text{i}) \land \text{hyp} \land \text{pre}
have "\text{dip } (\sigma i) \in kD (\text{rt } (\sigma (\text{zip } (\sigma i))))" \land \text{dsn } (\sigma i) \leq \text{nsq}n (\text{rt } (\sigma (\text{zip } (\sigma i)))) (\text{dip } (\sigma i)) \land (\text{dsn } (\sigma i)) = \text{dsn } (\sigma i) \rightarrow \text{the } (\text{dhop } (\text{rt } (\sigma (\text{zip } (\sigma i)))) (\text{dip } (\sigma i))) \leq \text{hop } (\sigma i) \lor \text{the } (\text{flag } (\text{rt } (\sigma (\text{zip } (\sigma i)))) (\text{dip } (\sigma i))) = \text{inv}''
by (auto simp: \{\text{zip } (\sigma i) = \text{i}\})
ultimatively show \?thesis
by (rule other quality_increases rreq rrep props)
next
assume "\text{zip } (\sigma i) = \text{i}" thus \?thesis
using \{\text{zip } (\sigma i) = \text{i}\} \land \text{hyp} \land \text{pre} by auto
qed

(auto simp: \{\text{seqlsimp elim!: quality_increases rreq rrep props\}})

lemma verr_sip:
"\text{opao}d i \Rightarrow (\text{other quality_increases } \{i\} (\text{orecvmsg msg_fresh}), \text{other quality_increases } \{i\} \Rightarrow)
\text{onl } \Gamma_{AODV} (\lambda(\sigma, i)).
\text{r1 } \in \{\text{PAoV}v : 8, \text{PAoV}d : 9, \text{PRrep} : 0, \text{PRrep} : 1\}
\rightarrow (\forall \text{rip} \in \text{dom } \text{dests } (\sigma i)). \text{rip} \in kD (\text{rt } (\sigma (\text{zip } (\sigma i)))) \land (\text{the } (\text{dests } (\sigma i) \text{ rip}) - 1 \leq \text{nsq}n (\text{rt } (\sigma (\text{zip } (\sigma i)))) \text{ rip})))" (is 
"?= (\text{?S, ?U} \Rightarrow =)

proof -
\{\text{fix } \text{dests } \text{rip} \text{ rsn } \text{and } \sigma \sigma' :: "ip = state"
assume quant: "\forall j. \text{quality_increases } (\sigma j) (\sigma' j)"
and *: "\forall \text{rip} \in \text{dom } \text{dests}. \text{rip} \in kD (\text{rt } (\sigma \text{ sip}))
\land (\text{the } (\text{dests } \text{ rip}) - 1 \leq \text{nsq}n (\text{rt } (\sigma \text{ sip})) \text{ rip}"
and "\text{dests } \text{ rip } = \text{ Some } \text{ rsn}"
from this(3) have "\text{rip} \in \text{dom } \text{dests}" by auto
with * and "\text{dests } \text{ rip } = \text{ Some } \text{ rsn}" have "\text{rip} \in kD (\text{rt } (\sigma \text{ sip}))"
and "\text{rsn} - 1 \leq \text{nsq}n (\text{rt } (\sigma \text{ sip})) \text{ rip}"
by (auto dest!: bspec)
from quant have "\text{quality_increases } (\sigma \text{ sip}) (\sigma' \text{ sip})" ..
have "rip \in kD(rt (\sigma' sip)) \land 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 equality\_increases\_nsqn\_fresh aodv\_wf aodv\_trans]
onl\_oinvariant\_sterms [OF aodv\_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: "paodv i \models
onl \Gamma_{AODV} (\lambda(\xi, l). (l = PRerr-1
\iff (\forall ip \in dom(dests \xi). ip \in vD(rt \xi)
\land the (nhop (rt \xi) ip) = sip \xi
\land 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\_aodv aodv\_trans aodv\_trans,
simplified seql\_onl\_swap,
THEN oinvariant\_anyact]

lemmas odests\_vD\_inc\_sqn =
open\_seq\_invariant [OF dests\_vD\_inc\_sqn initiali\_aodv aodv\_trans aodv\_trans,
simplified seql\_onl\_swap,
THEN oinvariant\_anyact]

lemmas oprerr\_guard =
open\_seq\_invariant [OF prerr\_guard initiali\_aodv aodv\_trans aodv\_trans,
simplified seql\_onl\_swap,
THEN oinvariant\_anyact]

Proposition 7.28

lemma seq\_compare\_next\_hop':
"opaodv i \models (\otherwith quality\_increases \{i\} (orecvmsg msg\_fresh),
other\_quality\_increases \{i\} \rightarrow onl \Gamma_{AODV} (\lambda(\sigma, _).
\forall dip. let nhip = the (nhop (rt \sigma dip))
in dip \in kD(rt (\sigma dip)) \land nhip \neq dip \rightarrow
dip \in kD(rt (\sigma nhip)) \land nsqn (rt (\sigma dip)) dip \leq nsqn (rt (\sigma nhip)) dip)"
(is ", _ \models (?S, ?U \rightarrow _)"
)

proof -

{ fix nhop and \sigma \sigma' :: "ip \Rightarrow state"
assume pre: "\forall dip \in kD(rt (\sigma dip)). nhop dip \neq dip \rightarrow
dip \in kD(rt (\sigma (nhop dip))) \land nsqn (rt (\sigma dip)) 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 dip)). nhop dip \neq dip \rightarrow
dip \in kD(rt (\sigma' (nhop dip))) \land nsqn (rt (\sigma dip)) dip \leq nsqn (rt (\sigma' (nhop dip))) dip"
proof (intro ballI impI)
fix dip
assume "dip \in kD(rt (\sigma dip))"
and "nhop dip \neq dip"
with pre have "dip \in kD(rt (\sigma (nhop dip)))"
and "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip" by auto

from qinc have qinc_nh: "quality_increases (σ (nhop dip)) (σ' (nhop dip))" ..
with \(\mu \in kD(\text{rt} (σ (nhop dip)))\) have "\(\mu \in kD (\text{rt} (σ' (nhop dip)))\)" ..

moreover have "nsqn (rt (σ i)) dip ≤ nsqn (rt (σ’ (nhop dip))) dip"
proof -
from \(\mu \in kD(\text{rt} (σ (nhop dip)))\) qinc_nh
have "nsqn (rt (σ (nhop dip))) dip ≤ nsqn (rt (σ’ (nhop dip))) dip" ..
with \(\mu \in nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (nhop dip))) dip\) show ?thesis
by simp
qed

ultimately show "\(\mu \in kD(\text{rt} (σ’ (nhop dip)))\)
\(\land nsqn (rt (σ i)) dip ≤ nsqn (rt (σ’ (nhop dip))) dip\)" ..
qed

} note basic = this

\{ fix nhop and \(σ, σ' :: \ "ip ⇒ state"\)
assume pre: "\(\forall \mu \in kD (\text{rt} (σ i))\). nhop dip \(\neq dip \rightarrow kD (\text{rt} (σ (nhop dip))) \land nsqn (rt (σ i)) dip \leq nsqn (rt (σ (nhop dip))) dip\)"
and ndest: "\(\forall ripc ∈ dom (dests (σ i)). ripc ∈ kD (\text{rt} (σ (sip (σ i)))) \land the (dests (σ i)) ripc) - 1 \leq nsqn (rt (σ (sip (σ i)))) ripc\)"
and issip: "\(\forall ip ∈ dom (dests (σ i)). nhop ip = sip (σ i)\)"
and qinc: "\(∀ j. quality_increases (σ j) (σ' j)\)"

have "\(\forall \mu \in kD(\text{rt} (σ i))\). nhop dip \(\neq dip \rightarrow kD (\text{rt} (σ’ (nhop dip)))\)
\(\land nsqn (invalidate (rt (σ i)) (dests (σ i))) dip \leq nsqn (rt (σ’ (nhop dip))) dip\)"
proof (intro ballI impI)
fix dip
assume "\(\mu \in kD(\text{rt} (σ i))\)"
and "\(\mu \neq dip\)"
with pre and qinc have "\(\mu \in kD(\text{rt} (σ’ (nhop dip)))\)"
and "\(nsqn (rt (σ i)) dip \leq nsqn (rt (σ’ (nhop dip))) dip\)" by (auto dest!: basic)

have "nsqn (invalidate (rt (σ i)) (dests (σ i))) dip \leq nsqn (rt (σ’ (nhop dip))) dip"
proof (cases "\(\mu \in dom (dests (σ i))\)"
assume "\(\mu \in dom (dests (σ i))\)"
with \(\mu \in kD(\text{rt} (σ i))\) obtain dsn where "\(\mu \in dom (dests (σ i)) dip = Some dsn\)"
by auto
with \(\mu \in kD(\text{rt} (σ i))\) have "\(nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = dsn - 1\)"
by (rule nsqn_invalidate_eq)
moreover have "\(dsn - 1 \leq nsqn (rt (σ’ (nhop dip))) dip\)"
proof -
from \(\mu \in dom (dests (σ i))\) have "\(the (dests (σ i)) dip = dsn\)" by simp
with ndest and \(\mu \in dom (dests (σ i))\) have "\(\mu \in kD (\text{rt} (σ (sip (σ i))))\)"
\(dsn - 1 \leq nsqn (rt (σ (sip (σ i)))) dip\)"
by auto
moreover from issip and \(\mu \in dom (dests (σ i))\) have "\(\mu \in kD (\text{rt} (σ (nhop dip)))\)"
ultimately have "\(\mu \in kD (\text{rt} (σ (nhop dip)))\)"
and "\(dsn - 1 \leq nsqn (rt (σ (nhop dip))) dip\)" by auto
with qinc show "\(dsn - 1 \leq nsqn (rt (σ’ (nhop dip))) dip\)"
by simp (metis kD nsqn_quality_increases_trans)
qed
ultimately show ?thesis by simp
next
assume "\(\mu \notin dom (dests (σ i))\)"
with \(\mu \in kD (\text{rt} (σ i))\)
have "\(nsqn (invalidate (rt (σ i)) (dests (σ i))) dip = nsqn (rt (σ i)) dip\)"
by (rule nsqn_invalidate_other)
with \(\mu \in nsqn (rt (σ i)) dip \leq nsqn (rt (σ’ (nhop dip))) dip\) show ?thesis by simp
qed

with \(\mu \in kD (\text{rt} (σ’ (nhop dip)))\)
show "\(\mu \in kD (\text{rt} (σ’ (nhop dip)))\)"
\[ \land \text{nsqn} (\text{invalidate} (\text{rt} (\sigma i)) (\text{dests} (\sigma i))) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop dip}))) \text{ dip} \] ..

qed

} note basic_prerr = this

}\fix \sigma \sigma' :: "ip \Rightarrow state"

assume a1: "\forall \text{dip} \in kD(\text{rt} (\sigma i)). \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}) \neq \text{dip}

\rightarrow \text{dip} \in kD(\text{rt} (\sigma (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}))))

\land \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 kD(\text{rt} (\sigma i)).

\text{the} (\text{nhop} (\text{update} (\text{rt} (\sigma i))) (0, \text{unk}, \text{val}, \text{Suc 0}, \text{sip} (\sigma i), \{})) \text{ dip} \neq \text{dip}

\rightarrow \text{dip} \in kD(\text{rt} (\sigma' (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}))))

\land \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 - (\text{drule(1) basic, auto})

thus "\forall \text{dip}" by (cases "\text{dip} = \text{sip} (\sigma i)") auto

qed

} note nhop_update_sip = this

}\fix \sigma \sigma' oip sip osn hops

assume pre: "\forall \text{dip} \in kD (\text{rt} (\sigma i)). \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}) \neq \text{dip}

\rightarrow \text{dip} \in kD(\text{rt} (\sigma (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}))))

\land \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 kD(\text{rt} (\sigma \text{ sip})))

\land \text{osn} \leq \text{nsqn} (\text{rt} (\sigma \text{ sip})) \text{ oip}

\land (\text{nsqn} (\text{rt} (\sigma \text{ sip})) \text{ oip} = \text{osn}

\rightarrow (\text{dhops} (\text{rt} (\sigma \text{ sip})) \text{ oip}) \leq \text{hops}

\lor (\text{the} (\text{flag} (\text{rt} (\sigma \text{ sip})) \text{ oip}) = \text{inv})"

from pre and qinc

have pre': "\forall \text{dip} \in kD (\text{rt} (\sigma i)). \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}) \neq \text{dip}

\rightarrow \text{dip} \in kD(\text{rt} (\sigma' (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}))))

\land \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 hops}, \text{sip}, \{})) \text{ oip}) \neq \text{oip}

\rightarrow \text{oip} \in 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{ oip})

\land \text{nsqn} (\text{update} (\text{rt} (\sigma i))) \text{ oip} (\text{osn}, \text{kno}, \text{val}, \text{Suc 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 hops}, \text{sip}, \{}))) \text{ oip}))"

(is "$\sigma' \text{nhop not oip} \rightarrow ?\text{oip in kD} \land ?\text{nsqn le nsqn}"

proof (rule, split update rt split_asm)

assume "rt (\sigma i) \neq \text{update} (rt (\sigma i)) \text{ oip} (\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip}, \{}"

and "the (\text{nhop} (rt (\sigma i)) \text{ oip}) \neq \text{oip}"

with pre' show "$?\text{oip in kD} \land ?\text{nsqn le nsqn}" by auto

next

assume rtnot: "rt (\sigma i) \neq \text{update} (rt (\sigma i)) \text{ oip} (\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip}, \{}"

and notoitp: ?nhop_not_oip

with * qinc have ?oip in kD

by (clarsimp elim!: kD quality increases)

moreover with * pre qinc rtnot notoitp have ?nsqn le nsqn

by simp (metis kD nsqn quality increases trans)

ultimately show "$?\text{oip in kD} \land ?\text{nsqn le nsqn}" ..
\document{qed}{note update1 = this}\\
{\setminus fix } σ σ' oip sip osn hops\\
\begin{align*}
&{\text{assume pre: }} "∀ \text{dip} ∈ kD (rt (σ i)). \text{the (nhop (rt (σ i)) dip) ≠ dip} \\
&{\quad → dip} ∈ kD (rt (σ (the (nhop (rt (σ i)) dip)))) \\
&{\quad ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip"}
&{\text{and qinc: }} "∀ j. quality_increases (σ j) (σ' j)"
&{\text{and *: }} "sip ≠ oip → oip ∈ kD (rt (σ sip)) \\
&{\quad ∧ osn ≤ nsqn (rt (σ sip)) oip \\
&{\quad ∧ (nsqn (rt (σ sip)) oip = osn} \\
&{\quad → the (dhops (rt (σ sip)) oip) ≤ hops} \\
&{\quad ∨ the (flag (rt (σ sip)) oip) = inv)"
\end{align*}
\begin{align*}
&{\text{from pre and qinc}} \\
&{\text{have pre': }} "∀ \text{dip} ∈ kD (rt (σ i)). \text{the (nhop (rt (σ i)) dip) ≠ dip} \\
&{\quad → dip} ∈ kD (rt (σ' (the (nhop (rt (σ i)) dip)))) \\
&{\quad ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ' (the (nhop (rt (σ i)) dip)))) dip"
&{\text{by (rule basic)}} \\
&{\text{have "∀ \text{dip} ∈ kD (rt (σ i)).} \\
&{\quad the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip} \\
&{\quad → dip} ∈ kD (rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip))) \\
&{\quad ∧ nsqn (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip} \\
&{\quad ≤ nsqn (rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip)))) dip"
&{\text{(is "∀ \text{dip} ∈ kD (rt (σ i)). } _ → ?dip_in_kD dip ∧ ?nsqn_le_nsqn dip")}
\end{align*}
\begin{align*}
&{\text{proof (intro ballI impI, split update_rt_split_asm)}} \\
&{\text{fix dip}} \\
&{\text{assume "dip} ∈ kD (rt (σ i))" \\
&{\quad and "the (nhop (rt (σ i)) dip) ≠ dip"} \\
&{\quad and "rt (σ i) = update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"
&{\text{with pre' show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip" by simp}}
\end{align*}
\begin{align*}
&{\text{next}} \\
&{\text{fix dip}} \\
&{\text{assume "dip} ∈ kD (rt (σ i))" \\
&{\quad and notdip: "the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip"} \\
&{\quad and rtnot: "rt (σ i) ≠ update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})"
&{\text{show "?dip_in_kD dip ∧ ?nsqn_le_nsqn dip"}}
\end{align*}
\begin{align*}
&{\text{proof (cases "dip = oip")}} \\
&{\text{assume "dip ≠ oip"}} \\
&{\text{with pre' (dip} ∈ kD (rt (σ i))) notdip} \\
&{\text{show ?thesis by clarsimp}}
\end{align*}
\begin{align*}
&{\text{next}} \\
&{\text{assume "dip = oip"}} \\
&{\text{with rtnot qinc (dip} ∈ kD (rt (σ i))) notdip * \\
&{\text{have "?dip_in_kD dip"}} \\
&{\text{by simp (metis kD_quality_increases)}} \\
&{\text{moreover from (dip = oip) rtnot qinc (dip} ∈ kD (rt (σ i))) notdip * \\
&{\text{have "?nsqn_le_nsqn dip" by simp (metis kD_nsqn_quality_increases_trans)}} \\
&{\text{ultimately show ?thesis ..}}
\end{align*}
\begin{align*}
&{\text{qed}}
\end{align*}
\begin{align*}
&{\text{note update2 = this}}
\end{align*}
\begin{align*}
&{\text{have "opaodv i |= (?S, ?U →) onl Γ_AODV (∑(σ, _).}} \\
&{\quad ∀ \text{dip} ∈ kD (rt (σ i)). \text{the (nhop (rt (σ i)) dip) ≠ dip} \\
&{\quad → dip} ∈ kD (rt (σ (the (nhop (rt (σ i)) dip)))) \\
&{\quad ∧ nsqn (rt (σ i)) dip ≤ nsqn (rt (σ (the (nhop (rt (σ i)) dip)))) dip)"
&{\text{by (inv_cterms inv add: oseq_step_invariant_sterms [OF oquality_increases_nsqn_fresh aodv_wf oaoadv_trans]}} \\
&{\text{onl_oinvariant_sterms [OF aodv_wf oaddpreRT_welldefined]}} \\
&{\text{onl_oinvariant_sterms [OF aodv_wf odests_vD_inc_sqn]}} \\
&{\text{onl_oinvariant_sterms [OF aodv_wf oprerr_guard]}}
\end{align*}
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_aodv,
simplified seql_onl_swap]

lemmas ozero_seq_unk_hops_one =
open_seq_invariant [OF zero_seq_unk_hops_one initiali_aodv,
simplified seql_onl_swap]

lemma reachable_fresh_okD_unk_or_atleast_one:
  fixes dip
  assumes "((σ, p) ∈ reachable (opaodv 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(rt (σ i) dip)) = unk ∨ 1 ≤ π2(rt (σ i) 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 okD_unk_or_atleast_one [OF oaodv_trans aodv_trans]],
              auto dest!: otherwith_actionD onlD simp: seqlsimp)
    with ⟨dip∈kD(rt (σ i))⟩ show ?thesis by simp
  qed

lemma reachable_fresh_ozero_seq_unk_hops_one:
  fixes dip
  assumes "((σ, p) ∈ reachable (opaodv 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 nhip = the (nhop (rt (σ i) dip))
in dip ∈ vD (rt (σ i)) \cap vD (rt (σ nhip)) \\
\wedge nhip \neq dip \\
\rightarrow (rt (σ i)) \sqsubseteq dip (rt (σ nhip)))

(is "\_ \models (\forall S i, \_ \rightarrow \_)")

proof -

have weaken:
"\forall I \ Q \ R \ P. \ p \models (otherwith quality_increases I (orecmsg Q), other quality_increases I \rightarrow) P \\
\Rightarrow p \models (otherwith (op=) I (orecmsg (\lambda \sigma. Q \sigma m \wedge R \sigma m)), other quality_increases I \rightarrow) P"

by auto

{ fix a and σ σ' :: "\textit{ip} \Rightarrow \textit{state}"

assume a1: "\forall dip. dip \in vD (rt (σ i)) \\
\wedge dip \in vD (rt (σ (the (nhop (rt (σ i)) dip)))) \\
\wedge (the (nhop (rt (σ i)) dip)) \neq dip \\
\rightarrow rt (σ i) \sqsubseteq dip \ rt (σ (the (nhop (rt (σ i)) dip)))"

and ow: "\forall S i σ σ' a"

have "\forall dip. dip \in vD (rt (σ i)) \\
\wedge dip \in vD (rt (σ' (the (nhop (rt (σ i)) dip)))) \\
\wedge (the (nhop (rt (σ i)) dip)) \neq dip \\
\rightarrow rt (σ i) \sqsubseteq dip \ rt (σ' (the (nhop (rt (σ i)) dip)))"

proof clarify

fix dip

assume a2: "\textit{dip} \in vD (rt (σ i))"

and a3: "\textit{dip} \in vD (rt (σ' (the (nhop (rt (σ i)) dip))))"

and a4: "(the (nhop (rt (σ i)) dip)) \neq dip"

from ow have "\forall j. \ j \neq i \rightarrow σ j = σ' j" by auto

show "rt (σ i) \sqsubseteq 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 \textit{dip} \in vD (rt (σ i)) have "\textit{dip} \in vD (rt (σ (the (nhop (rt (σ i)) dip))))" by simp

with a1 a2 a4 have "rt (σ i) \sqsubseteq dip \ rt (σ (the (nhop (rt (σ i)) dip)))" by simp

with (the (nhop (rt (σ i)) dip)) = i have "rt (σ i) \sqsubseteq dip \ rt (σ i)" by simp

hence False by simp

thus \textit{?thesis} ..

next

assume "(the (nhop (rt (σ i)) dip)) \neq i"

with \forall j. j \neq i \rightarrow σ j = σ' j

have "σ (the (nhop (rt (σ i)) dip)) = σ' (the (nhop (rt (σ i)) dip))" by simp

with d \textit{dip} \in vD (rt (σ' (the (nhop (rt (σ i)) dip)))) have "\textit{dip} \in vD (rt (σ (the (nhop (rt (σ i)) dip))))" by simp

with a1 a2 a4 have "rt (σ i) \sqsubseteq dip \ rt (σ (the (nhop (rt (σ i)) dip)))" by simp

with * show \textit{?thesis} by simp

qed

qed

} note basic = this

{ fix σ σ' a dip sip i

assume a1: "\forall dip. dip \in vD (rt (σ i)) \\
\wedge dip \in vD (rt (σ (the (nhop (rt (σ i)) dip)))) \\
\wedge (the (nhop (rt (σ i)) dip)) \neq dip \\
\rightarrow rt (σ i) \sqsubseteq dip \ rt (σ (the (nhop (rt (σ i)) dip)))"

and ow: "\forall S i σ σ' a"

have "\forall dip. dip \in vD (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) \\
\wedge dip \in vD (rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip)))) \\
\wedge (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip)) \neq dip \\
\rightarrow update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {}) \\
\sqsubseteq dip \ rt (σ' (the (nhop (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})) dip)))"

proof clarify

fix dip

assume a2: "\textit{dip} \in vD (update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {}))"

and a3: "\textit{dip} \in 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)) \neq dip"

show "update (rt (σ i)) sip (0, unk, val, Suc 0, sip, {})"
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)) \or 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)) \wedge dip ∈ \vD\ (σ (nhop dip)) \wedge nhop dip ≠ dip
    \or 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)))
    \wedge dip ∈ \vD\ (σ' (nhop dip)) \wedge nhop dip ≠ dip
    \or 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\ (σ' (nhop dip))"
    and "nhop dip ≠ dip"
    from this(I) have "dip ∈ \vD\ (rt (σ i))"
      by (clarsimp dests!: vD_invalidate_vD_not_dests)
    moreover from ow have "∀ j. j ≠ i \or σ j = σ' j" by auto
    ultimately have "rt (σ i) ⊆ dip rt (σ' (nhop dip))"
      using pre \{dip ∈ \vD\ (rt (σ' (nhop dip))) \wedge nhop dip ≠ dip\}
      by metis
    with \{∀ j. j ≠ i \or 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))
    \wedge dip ∈ \vD\ (σ (the (nhop (rt (σ i)) dip)))
    \wedge the (nhop (rt (σ i)) dip) ≠ dip
    \or 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 \or oip ∈ \kD\ (rt (σ sip))
    \wedge osn ≤ nsqn (rt (σ sip)) oip
    \wedge (nsqn (rt (σ sip)) oip = osn
      \or the (dhops (rt (σ sip)) oip) ≤ hops
      \wedge 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, {}))
    \wedge dip ∈ \vD\ (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip)))
    \wedge the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip) ≠ dip
    \or update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})
    \or rt (σ' (the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip, {})) dip)))"
    by (clarsimp dest!: vD_invalidate_vD_not_dests)
  qed
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 ∈ kD (rt (σ i))" by simp
moreover from a3 have "dip ∈ kD (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 from 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 ∈ vD (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
    hence "("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 simp: 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

} note rreq_rrep_update = this

have "opaodv 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 aodv_wf rreq_sip [THEN weaken]]

onl_oinvariant_sterms [OF aodv_wf rrep_sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf rrrep sip [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf oosn_rreq [THEN weaken]]
onl_oinvariant_sterms [OF aodv_wf odsn_rrep [THEN weaken]]
onl_oinvariant_sterms [OF aodv_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{opaodv} i) (\text{?S} i) (\text{other quality_increases} \{i\})" and "other quality_increases \{i\} \( \sigma \), \( \sigma' \)"
and 11: "\( l \in \text{labels} \Gamma_{AODV} \)
and pre: "\( \forall \text{dip. dip} \in \text{vD} (\text{rt} (\sigma i)) \land \text{dhops} (\text{rt} (\sigma i)) \text{ dip} = \text{unm} \land \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}) \neq \text{dip} \)
\rightarrow \text{rt} (\sigma i) \sqsubseteq \text{dip} (\text{rt} (\sigma (\text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip})))""

from this(1-2)
have or': "\((\sigma', p) \in \text{oreachable} (\text{opaodv} i) (\text{?S} i) (\text{other quality_increases} \{i\})" by (rule oreachable_other')

from or and 11 have next_hop: "\( \forall \text{dip. let nhip = the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}) \in \text{dip} \in kD(\text{rt} (\sigma i)) \land \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}) \neq \text{dip} \land \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機關_next_hop'])

from or and 11 have unk_hops_one: "\( \forall \text{dip. dip} \in kD (\text{rt} (\sigma i)). \text{sqn} (\text{rt} (\sigma i)) \text{ dip} = 0 \)
\rightarrow \text{sqnf} (\text{rt} (\sigma i)) \text{ dip} = \text{unm} \land \text{the} (\text{dhops} (\text{rt} (\sigma i)) \text{ dip}) = 1 \land \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 aodv_trans aodv_trans] otherwith_actionD simp: seqsimp)

from (other quality_increases \{i\} \( \sigma \), \( \sigma' \)) have "\( \sigma' i = \sigma i \)" 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 "\( \forall \text{dip. dip} \in \text{vD} (\sigma' i) \land \text{dip} \in kD(\text{rt} (\sigma' i)) \land \text{the} (\text{nhop} (\text{rt} (\sigma' i)) \text{ dip}) \neq \text{dip} \)
\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 "\( \text{dip} \in \text{vD}(\text{rt} (\sigma' i)) \)"
and "\( \text{dip} \in \text{vD}(\text{rt} (\sigma' (\text{the} (\text{nhop} (\text{rt} (\sigma' i)) \text{ dip}))))) \)"
and "\( \text{the} (\text{nhop} (\text{rt} (\sigma' i)) \text{ dip}) \neq \text{dip} \)"

from this(1) and \( \sigma' i = \sigma i \) have "\( \text{dip} \in \text{vD}(\text{rt} (\sigma i)) \)"
and "\( \text{dip} \in kD(\text{rt} (\sigma i)) \)"

by auto

from (\( \text{the} (\text{nhop} (\text{rt} (\sigma' i)) \text{ dip}) \neq \text{dip} \)) and \( \sigma' i = \sigma i \)
have "\( \text{the} (\text{nhop} (\text{rt} (\sigma i)) \text{ dip}) \neq \text{dip} \) (is "\( ?\text{nhip} \neq _\)") by simp
with \( \text{dip} \in kD(\text{rt} (\sigma i)) \) and next_hop
have "\( \text{dip} \in kD(\text{rt} (\sigma (?\text{nhip}))) \)"
and nsqns: "\( \text{nsqn} (\text{rt} (\sigma i)) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma ?\text{nhip})) \text{ dip} \)"
by (auto simp: Let_def)

have "\( 0 < \text{sqn} (\text{rt} (\sigma i)) \text{ dip} \)"
proof (rule neq0_conv [THEN iffD1, OF notI])
assume "\( \text{sqn} (\text{rt} (\sigma i)) \text{ dip} = 0 \)"
with \( \text{dip} \in kD(\text{rt} (\sigma i)) \) and unk_hops_one
have "?\text{nhip} = \text{dip}" by simp

243
with \( ?\text{nhip} \neq \text{dip} \) show False ..

\[ \text{qed} \]

also have "\( \ldots = \text{nsqn} (\text{rt} (\sigma \ i)) \text{dip} \)"
by \( \text{rule vD_nsqn_sqn [OF } \text{dip} \in \text{vD} (\text{rt} (\sigma \ i)), \text{THEN sym]} \)
also have "\( \ldots \leq \text{nsqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{dip}) \)"
by \( \text{rule nsqns} \)
also have "\( \ldots \leq \text{sqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{dip}) \)"
by \( \text{rule nsqn_sqn} \)
finally have "\( 0 < \text{sqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{dip}) \)".

have "\( \text{rt} (\sigma \ i) \sqsubseteq \text{dip} \ \text{rt} (\sigma \ ?\text{nhip}) \)"
proof (cases "\( \text{dip} \in \text{vD} (\text{rt} (\sigma \ ?\text{nhip})) \)"
assume "\( \text{dip} \in \text{vD} (\text{rt} (\sigma \ ?\text{nhip})) \)"
with \( \text{pre} \ (\text{dip} \in \text{vD} (\sigma \ i)) \) and \( ?\text{nhip} \neq \text{dip} \)
have "\( \text{rt} (\sigma \ i) \sqsubseteq \text{dip} \ (\sigma \ ?\text{nhip}) \)" by auto
moreover from \( \forall j. \text{quality_increases} (\sigma \ j) (\sigma \ ?\text{nhip}) \)
have "\( \text{quality_increases} (\sigma \ ?\text{nhip}) (\sigma \ ?\text{nhip}) \)"
ultimately show \( \text{thesis} \)
using \( \text{dip} \in \text{kD} (\text{rt} (\sigma \ ?\text{nhip})) \)
by \( \text{rule strictly_fresher_quality_increases_right} \)
next
assume "\( \text{dip} \in \text{vD} (\text{rt} (\sigma \ ?\text{nhip})) \)"
with \( \text{dip} \in \text{kD} (\text{rt} (\sigma \ ?\text{nhip})) \) have "\( \text{dip} \in \text{iD} (\text{rt} (\sigma \ ?\text{nhip})) \)"
by \( \text{rule nsqns} \)
also from \( \text{dip} \in \text{iD} (\text{rt} (\sigma \ ?\text{nhip})) \)
have "\( \ldots = \text{nsqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{dip}) - 1 \)"
also have "\( \ldots < \text{sqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{dip}) \)"
proof -
from \( \forall j. \text{quality_increases} (\sigma \ j) (\sigma \ ?\text{nhip}) \)
have "\( \text{quality_increases} (\sigma \ ?\text{nhip}) (\sigma \ ?\text{nhip}) \)"
hence "\( \forall \text{ip}. \text{sqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{ip}) \leq \text{sqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{dip}) \)" by auto
hence "\( \text{sqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{dip}) \leq \text{sqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{dip}) \)".
with \( 0 < \text{sqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{dip}) \) show \( \text{thesis} \) by auto
\[ \text{qed} \]
also have "\( \ldots = \text{nsqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{dip}) \)"
proof \( \text{rule vD_nsqn_sqn [THEN sym]} \)
from \( \text{dip} \in \text{vD} (\text{rt} (\sigma \ (\text{the} \ (\text{nhop} (\text{rt} (\sigma \ i)) \text{dip})))) \) and \( \sigma \ i = \sigma \ i \)
show "\( \text{dip} \in \text{vD} (\text{rt} (\sigma \ ?\text{nhip})) \)" by simp
\[ \text{qed} \]
finally have "\( \text{nsqn} (\text{rt} (\sigma \ i)) \text{dip} < \text{nsqn} (\text{rt} (\sigma \ ?\text{nhip}) \text{dip}) \)".

moreover from \( \text{dip} \in \text{vD} (\text{rt} (\sigma \ (\text{the} \ (\text{nhop} (\text{rt} (\sigma \ i)) \text{dip})))) \) and \( \sigma \ i = \sigma \ i \)
have "\( \text{dip} \in \text{kD} (\text{rt} (\sigma \ ?\text{nhip})) \)" by auto
ultimately show "\( \text{rt} (\sigma \ i) \sqsubseteq \text{dip} \ (\sigma \ ?\text{nhip}) \)"
using \( \text{dip} \in \text{kD} (\text{rt} (\sigma \ i)) \) by \( \text{rule rt_strictly_fresher_ltI} \)
\[ \text{qed} \]
with \( \sigma \ i = \sigma \ i \) show "\( \text{rt} (\sigma \ i) \sqsubseteq \text{dip} \ (\sigma \ (\text{the} \ (\text{nhop} (\text{rt} (\sigma \ i)) \text{dip})))) \)"
by simp
\[ \text{qed} \]
\[ \text{thus } \text{thesis unfolding Let_def} . \]
\[ \text{qed} \]

lemma seq_compare_next_hop:
fixes \( w \)
shows "\( \text{opaodv} \ i \vdash (\text{otherwith} \ (\text{op} = \{i\}) \ \text{(orecvmsg msg_fresh)}, \)
other quality_increases \( \{i\} \rightarrow) \)
global (\( \lambda \sigma. \ \forall \text{dip}. \ \text{let} \ nhip = \text{the} \ (\text{nhop} (\text{rt} (\sigma \ i)) \text{dip}) \)
in \( \text{dip} \in \text{kD} (\text{rt} (\sigma \ i)) \) \( \land \ nhip \neq \text{dip} \rightarrow \)
dip \in \text{kD} (\text{rt} (\sigma \ nhip))
\( \land \text{nsqn} (\text{rt} (\sigma \ i)) \text{dip} \leq \text{nsqn} (\text{rt} (\sigma \ nhip) \text{dip}) \)"
by (rule oinvariant_weakenE [OF seq_compare_next_hop']) (auto dest!: onlD)

lemma seq_nhop_quality_increases:
shows "opaodv i \|= (otherwith (op=) {i}
(orecmsg (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

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 ⇒ state) ⇒ ip ⇒ ip rel"
where
"rt_graph σ = (λdip. 
{(ip, ip') | ip ip' dsn dsk hops pre. 
  ip ≠ dip ∧ rt (σ ip) = Some (dsn, dsk, val, hops, ip', pre)})"
Given the state of a network σ, 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 = Some (dsn, dsk, val, hops, ip', pre))"
using assms unfolding rt_graph_def by auto

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

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

lemma rt_graph_not_dip_trans [dest]:
"\land ip ip' 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]:
"\land ip ip' 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) +)"

245
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)) \land nhip \neq dip \rightarrow rt (\sigma ip) \sqsubseteq 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) \sqsubseteq 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) \sqsubseteq 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: \[ \[ dip \in vD(rt (\sigma nhip)); nhip \neq dip \] \rightarrow rt (\sigma ip) \sqsubseteq 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) \sqsubseteq dip rt (\sigma nhip)" by (rule IH) 
also have "rt (\sigma nhip) \sqsubseteq dip rt (\sigma nhip')" 
proof - 
from dip \in vD(rt (\sigma nhip)) and dip \in vD(rt (\sigma nhip')) 
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) \sqsubseteq dip rt (\sigma nhip')" 
by (clarsimp simp: Let_def) 
qed 
finally show "rt (\sigma ip) \sqsubseteq 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) \sqsubseteq 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 σ s a σ' s'
  assumes "((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos Γ_AODV i) (seqp_sos Γ_QMSG)"
  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. 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 ∥⟨⟨ i qmsg |= (otherwith (op=) {i}) (orecvmsg (λσ. msg_fresh σ m ∧ msg_zhops m)), other quality_increases {i} →) global (λσ. ∀ dip = 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 ostep_invariant_weakenE [OF oquality_increases], simp_all)
    fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
    assume "onll Γ_AODV (λ((σ, _), _, (σ', _)). 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 (λ_. rreq_rrep_sn) σ, other (λ_ _. True) {i} →) globala (λ σ, _, σ'). quality_increases (σ i) (σ' i)"
    thus "otherwith quality_increases (σ 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 |= (otherwith (op=) {i}) (orecvmsg (λ_. orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →) globala (λ σ, _, σ'). quality_increases (σ i) (σ' i))"
  proof
    have "opaodv i ∥⟨⟨ i qmsg |= (otherwith (op=) {i}) (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], simp_all)
    hence "opaodv i ∥⟨⟨ i qmsg |= (otherwith (op=) {i}) (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:
  "opaodv i ∥⟨⟨ i qmsg |= (otherwith (op=) {i}) (orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →) globala (λ σ, a, σ'). anycast (msg_fresh σ a)"
  proof
    have "opaodv i ∥⟨⟨ i qmsg |= (otherwith (op=) {i}) (orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) {i} →) globala (λ σ, a, σ'). anycast (msg_fresh σ a)"
    by (rule ostep_invariant_weakenE [OF oreq_rreq_rrep_nsqn_fresh_any_step_invariant])
    fix t

247
assume "onll Γ_{AODV} (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 aodv_ex_label)
qed

hence "opaodv i ⟨⟨ {i} qmsg |=_ A (λσ _. orccvmsg (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 "opaodv 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_aodv oaoadv_trans aodv_trans
  have "opaodv i ⟨⟨ {i} qmsg |=_ A (act TT, other (λ _ _ _ _ True) {i} →) seqll i (onll Γ_{AODV} (λ(_, a, _). anycast msg_zhops a)) t"
    by (rule open_seq_step_invariant)
  hence "opaodv i ⟨⟨ {i} qmsg |=_ 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 Γ_{AODV} (λ(_, a, _). anycast msg_zhops a)) t"
      thus "globala (λ(_, a, _, _). anycast msg_zhops a) t"
        by (cases t) (clarsimp dest!: seqllD onllD, metis aodv_ex_label)
    qed simp_all
  hence "opaodv i ⟨⟨ {i} qmsg |=_ A (λσ __. orccvmsg (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

2.11.2 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 Γ_{AODV} i) (seqp_sos Γ_{QM SG}))" 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 : opaodv i ⟨⟨ {i} qmsg : R ⟩o |= (oparp_sos i (oseqp_sos Γ_{AODV} i) (seqp_sos Γ_{QMSG}))⟩
    (σ, NodeS i P R), a, (σ', NodeS i' P' R')) ∈ onode_sos (oparp_sos i (oseqp_sos Γ_{AODV} i) (seqp_sos Γ_{QMSG}))"
    ⟨op= i⟩ (oparp_sos i (oseqp_sos Γ_{AODV} i) (seqp_sos Γ_{QMSG}))" 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_quality_increases:
  shows "⟨ i : opaodv i ⟨⟨ {i} qmsg : R ⟩o |= (oparp_sos i (oseqp_sos Γ_{AODV} i) (seqp_sos Γ_{QMSG}))⟩
    (σ, NodeS i P R), a, (σ', NodeS i' P' R')) ∈ onode_sos (oparp_sos i (oseqp_sos Γ_{AODV} i) (seqp_sos Γ_{QMSG}))"
    ⟨op= i⟩ (oparp_sos i (oseqp_sos Γ_{AODV} i) (seqp_sos Γ_{QMSG}))" 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_rreq_rrep_nsqn_fresh_any_step:
  shows "⟨ i : opaodv i ⟨⟨ {i} qmsg : R ⟩o |= (oparp_sos i (oseqp_sos Γ_{AODV} i) (seqp_sos Γ_{QMSG}))⟩
    (σ, NodeS i P R), a, (σ', NodeS i' P' R')) ∈ onode_sos (oparp_sos i (oseqp_sos Γ_{AODV} i) (seqp_sos Γ_{QMSG}))"
    ⟨op= i⟩ (oparp_sos i (oseqp_sos Γ_{AODV} i) (seqp_sos Γ_{QMSG}))" 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_anycast_msg_zhops:
shows "(i : opaodv i (\{i qmsg : R\}o) \models
(\lambda \sigma_\_. oarrivemsg (\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 "(i : opaodv i (\{i qmsg : R\}o) \models
(\lambda \sigma_\_. oarrivemsg (\lambda_\_. True) \sigma, other (\lambda_\_. True) \{i\} \rightarrow)
globala (\lambda (\sigma, a, \sigma'). a \neq \tau \rightarrow \sigma' = \tau i = \sigma i)"
proof (rule ostep_invariantI, simp (no_asm), rule impI)
fix \sigma, \zeta, \zeta'
assume or: "(\sigma, \zeta) \in ooreachable ((i : opaodv i (\{i qmsg : R\}o))
(\lambda \sigma_\_. oarrivemsg (\lambda_\_. True) \sigma)
(other (\lambda_\_. True) \{i\})"
and tr: "((\sigma, \zeta), a, (\sigma', \zeta')) \in trans ((i : opaodv i (\{i qmsg : R\}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, i. NodeS i p R), a, (\sigma', \zeta'))
\in onode_sos (oparp_sos i (trans (opaodv 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

2.11.3 Lift to partial networks

lemma arrive_rreq_rrep_nsqn_fresh_inc_sn [simp]:
assumes "oarrivemsg (\lambda_\_. msg_fresh \sigma m \land P \sigma m) \sigma m"
shows "oarrivemsg (\lambda_. rreq_rrep_sn) \sigma m"
using assms by (cases m) auto

lemma opnet_nhops_quality_increases:
shows "opnet (\lambda i. opaodv i (\{i qmsg : R\}o) \models
(otherwith (op=) (net_tree_ips p)
(oarrivemsg (\lambda \sigma_\_. msg_fresh \sigma m \land msg_zhops m)),
other_quality_increases (net_tree_ips p) \rightarrow)
globala (\lambda \sigma. \forall i\in\net_tree_ips p. \forall dip.
let nhp = the (nhop (rt (\sigma i)) dip)
in dip \in vD (rt (\sigma i)) \land vD (rt (\sigma nhp)) \land nhp \neq dip
\rightarrow (rt (\sigma i)) \sqcup dip (rt (\sigma nhp)))"
proof (rule pnet_lift [OF node_nhops_quality_increases])
fix i R
have "(i : opaodv i (\{i qmsg : R\}o) \models
(\lambda \sigma_\_. oarrivemsg (\lambda_. rreq_rrep_sn) \sigma, other (\lambda_\_. True) \{i\} \rightarrow)
globala (\lambda (\sigma, a, \sigma').
castmsg (\lambda m. msg_fresh \sigma m \land msg_zhops m) a)"
proof (rule ostep_invariantI, simp (no_asm))
fix \sigma, a, \sigma', \sigma''
assume or: "(\sigma, a) \in ooreachable ((i : opaodv i (\{i qmsg : R\}o))
(\lambda \sigma_\_. oarrivemsg (\lambda_. rreq_rrep_sn) \sigma)
(other (\lambda_\_. True) \{i\})"
and tr: "((\sigma, a), (\sigma', \sigma'')) \in trans ((i : opaodv i (\{i qmsg : R\}o))"
and am: "oarrivemsg (\lambda_. 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. msg_fresh \sigma m \land msg_zhops m) a"
by (case_tac a) auto
qed
thus "(i : opaodv i (\{i qmsg : R\}o) \models
(\lambda \sigma_\_. oarrivemsg (\lambda \sigma m. msg_fresh \sigma m \land msg_zhops m) \sigma,
other quality_increases \{i\} \rightarrow \text{globa} \ (\lambda (\sigma, a, \_).
\text{castmsg} \ (\lambda m. \text{msg}_{\text{fresh}} \sigma m \land \text{msg}_{\text{zhops}} m) a)"

by rule auto

next

fix i R

show "\{i\} : \text{opaodv} i \{\{i \text{ qmsg} \ : \ R\}_o \models_A
(\lambda \sigma \_ . \text{oarrivemsg} \ (\lambda \sigma m. \text{msg}_{\text{fresh}} \sigma m \land \text{msg}_{\text{zhops}} m) \sigma,
\text{other quality_increases} \{i\} \rightarrow \text{globa} \ (\lambda (\sigma, a, \sigma').
\quad \quad \quad a \neq \tau \land (\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\} : \text{opaodv} i \{\{i \text{ qmsg} \ : \ R\}_o \models_A
(\lambda \sigma \_ . \text{oarrivemsg} \ (\lambda \sigma m. \text{msg}_{\text{fresh}} \sigma m \land \text{msg}_{\text{zhops}} m) \sigma,
\text{other quality_increases} \{i\} \rightarrow \text{globa} \ (\lambda (\sigma, a, \sigma').
\quad \quad \quad a = \tau \lor (\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

2.11.4 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)
\text{global} \ (\lambda \sigma. \forall i \in \text{net_tree_ips} p. \forall dip.
\quad \quad \quad \text{let nhip} = \text{the} (\text{nhop} (\text{rt} (\sigma i)) dip)
\quad \quad \quad \text{in dip} \in \text{vD} (\text{rt} (\sigma i)) \cap \text{vD} (\text{rt} (\sigma nhip)) \land nhip \neq \text{dip}
\quad \quad \quad \rightarrow (\text{rt} (\sigma i)) {\subseteq_{\text{dip}}} (\text{rt} (\sigma nhip)))"

(is "\models (\_, ?U \rightarrow) ?inv")

proof (rule inclosed_closed)

from opnet_nhop_quality_increases

show "opnet (\lambda i. \text{opaodv} i \langle\langle i \text{ qmsg} \rangle\rangle p)
\models (\text{otherwith} (\text{op=}) (\text{net_tree_ips} p) \text{ inoclosed}, \_ \rightarrow) ?inv"

proof (rule oinvariant_weakenE)

fix \sigma \sigma' :: \"ip \Rightarrow state\" and a :: \"msg node_action\"

assume "\text{otherwith} (\text{op=}) (\text{net_tree_ips} p) \text{ inoclosed} \ \sigma \ \sigma' \ a"

thus "\text{otherwith} (\text{op=}) (\text{net_tree_ips} p)
\text{oarrivemsg} \ (\lambda \sigma m. \text{msg}_{\text{fresh}} \sigma m \land \text{msg}_{\text{zhops}} m) \ \sigma \ \sigma' \ a"

proof (rule otherwithEI)

fix \sigma :: \"ip \Rightarrow state\" and a :: \"msg node_action\"

assume "\text{inoclosed} \ \sigma \ a"

thus "\text{oarrivemsg} \ (\lambda \sigma m. \text{msg}_{\text{fresh}} \sigma m \land \text{msg}_{\text{zhops}} m) \ \sigma \ a"

proof (cases a)

fix ii ni ms

assume "a = ii \rightarrow ni: \text{arrive}(ms)"

moreover with \"\text{inoclosed} \ \sigma \ a\" obtain d di where "ms = \text{newpkt}(d, di)"

by (cases ms) auto

ultimately show ?thesis by simp

qed simp_all

qed

2.11.5 Transfer into the standard model

interpretation \text{aodv_openproc}: \text{openproc} \text{ paodv} \text{ oadv} \text{ id}

where "\text{aodv_openproc.initmissing} = \text{initmissing}"

proof -

show "\text{openproc} \text{ paodv} \text{ oadv} \text{ id}"

proof unfold_locales

fix i :: ip

have "\{\sigma, \zeta\}. \ (\sigma i, \zeta i) \in \sigma_{AODV} i \land (\forall j. j \neq i \rightarrow \sigma j \in \text{fst}' \sigma_{AODV} j)\} \subseteq \sigma_{AODV}'"

unfolding \sigma_{AODV'}_def \sigma_{AODV'}'_def

proof (rule equalityD1)
show \( \forall p. \{ (\sigma, \zeta). (\sigma i, \zeta) \in \{(f i, p)\} \wedge (\forall j. j \neq i \rightarrow \sigma j \in \text{fst}' \{(f j, p)\}) = \{(f, p)\} \) \\
by (rule set_eqI) auto \\
qed \\
thus \( \{ (\sigma, \zeta) | (\sigma i, \zeta) = \text{id} s \wedge (\forall j. j \neq i \rightarrow \sigma j \in (\text{fst} o \text{id}) ' \text{init}(paodv j)) \} \subseteq \text{init}(opaodv i) \)
by simp
next 
show \( \forall j. \text{init}(paodv j) \neq \{\} \)
unfolding \( \sigma_{AODV}_\text{def} \) by simp
next 
fix \( i \) \( s \) a \( s' \) \( \sigma \) \( \sigma' \)
assume \( \sigma i = \text{fst}(\text{id} s) \) 
and \( \sigma' i = \text{fst}(\text{id} s') \) 
and \( (\sigma, a, \sigma') \in \text{trans}(paodv i) \)
then obtain \( q \) \( q' \) where \( s = (\sigma i, q) \)
and \( s' = (\sigma' i, q') \) 
and \( (((\sigma i, q), a, (\sigma' i, q')) \in \text{trans}(paodv i) \)
by (cases s, cases s') auto 
from this(3) have \( (((\sigma i, q), a, (\sigma' i, q')) \in \text{trans}(opaodv i) \)
by simp (rule open_seqp_action [OF aodv_wf])
with \( \langle s = (\sigma i, q) \rangle \) and \( \langle s' = (\sigma' i, q') \rangle \)
show \( (((\sigma, \text{snd}(\text{id} s)), a, (\sigma', \text{snd}(\text{id} s'))) \in \text{trans}(opaodv i) \)
by simp
qed 
then interpret op: openproc paodv opaodv id . 
have [simp]: \( \lambda i. (\text{SOME} x. x \in (\text{fst} o \text{id}) ' \text{init}(paodv i)) = \text{aodv_init} i \) 
unfolding \( \sigma_{AODV}_\text{def} \) by simp
hence \( \lambda i. \text{openproc.initmissing}(\text{paodv} i \langle\langle \text{qmsg}\rangle\langle\lambda (p, q). (\text{fst}(\text{id} p), \text{snd}(\text{id} p), q)\rangle\rangle) = \text{initmissing i} \)
by simp 
interpretation \( \text{aodv_openproc_par_qmsg}: \text{openproc_parq paodv opaodv id qmsg} \)
where \( \text{"aodv_openproc_par_qmsg.netglobal = netglobal"} \)
and \( \text{"aodv_openproc_par_qmsg.initmissing = initmissing"} \)
proof - 
show \( \text{"openproc_parq paodv opaodv id qmsg"} \)
by (unfold_locales) simp
then interpret \( \text{opq: openproc_parq paodv opaodv id qmsg} \).
have im: \( \lambda \sigma. \text{openproc.initmissing}(\lambda i. \text{paodv i} (\langle\langle \text{qmsg}\rangle\langle\lambda (p, q). (\text{fst}(\text{id} p), \text{snd}(\text{id} p), q)\rangle\rangle) \sigma = \text{initmissing} \sigma\) 
unfolding opq.initmissing_def opq.someinit_def initmissing_def 
unfolding \( \sigma_{AODV}_\text{def} \) \( \sigma_{QMSG}_\text{def} \) by (clarsimp cong: option.case_cong)
thus \( \text{"openproc.initmissing}(\lambda i. \text{paodv i} (\langle\langle \text{qmsg}\rangle\langle\lambda (p, q). (\text{fst}(\text{id} p), \text{snd}(\text{id} p), q)\rangle\rangle) P \sigma = \text{netglobal} P \sigma\) \)
unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def 
unfolding \( \sigma_{AODV}_\text{def} \) \( \sigma_{QMSG}_\text{def} \) 
by (clarsimp cong: option.case_cong)
simp del: One_nat_def 
simp add: fst_initmissing_netgmap_default_aodv_init_netlift 
[symmetric, unfolded initmissing_def])
thus \( \text{"openproc.netglobal}(\lambda i. \text{paodv i} (\langle\langle \text{qmsg}\rangle\langle\lambda (p, q). (\text{fst}(\text{id} p), \text{snd}(\text{id} p), q)\rangle\rangle) = \text{netglobal}" \)
by auto 
qed 

lemma net_nhop_quality_increases: 
assumes \( \text{"wf_net_tree n"} \)
shows "closed (pnet (λi. paodv 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. paodv 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 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 σ i
assume sr: "σ ∈ reachable (closed (pnet (λi. paodv i ⟨⟨ qmsg) n)) TT"

hence "∀ i∈net_tree_ips n. ?inv (fst (initmissing (netgmap fst σ))) i" by simp only: aodv_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. paodv 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 = aodv_init i" by simp
  thus ?thesis by simp
qed metis

qed

2.11.6 Loop freedom of AODV

theorem aodv_loop_freedom:
  assumes "wf_net_tree n"
shows "closed (pnet (λi. paodv i ⟨⟨ qmsg) n) ≜ netglobal (λσ. ∀ dip. irrefl ((rt_graph σ dip) +))"
using assms by (rule aodv_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 ⇒ sqn"
  where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ 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 ≠ Suc 0"
  unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 1"
  by simp

3.1.2 Modelling Routes

A route is a 5-tuple, (dsn, dsk, flag, hops, nhip) 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 nhip is the next hop toward the destination. In this variant, the set of ‘precursor nodes’ is not modelled.

type_synonym r = "sqn × k × f × nat × ip"
definition proj2 :: \"r \Rightarrow sqn\" (\"\pi_2\")
  where \"\pi_2 \equiv \lambda(ds, _, _, _, _). ds\"

definition proj3 :: \"r \Rightarrow k\" (\"\pi_3\")
  where \"\pi_3 \equiv \lambda(_, ds, _, _, _). ds\"

definition proj4 :: \"r \Rightarrow f\" (\"\pi_4\")
  where \"\pi_4 \equiv \lambda(_, _, flag, _, _). flag\"

definition proj5 :: \"r \Rightarrow nat\" (\"\pi_5\")
  where \"\pi_5 \equiv \lambda(_, _, _, hops, _). hops\"

definition proj6 :: \"r \Rightarrow ip\" (\"\pi_6\")
  where \"\pi_6 \equiv \lambda(_, _, _, nhip). nhip\"

lemma projs [simp]:
  \"\pi_2(ds, ds, flag, hops, nhip) = ds\"
  \"\pi_3(ds, ds, flag, hops, nhip) = ds\"
  \"\pi_4(ds, ds, flag, hops, nhip) = flag\"
  \"\pi_5(ds, ds, flag, hops, nhip) = hops\"
  \"\pi_6(ds, ds, 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 ] \implies P (\pi_3 x)\"
  by (rule k.induct)

lemma proj4_pred [intro]: \"[ P val; P inv ] \implies P (\pi_4 x)\"
  by (rule f.induct)

lemma proj6_pair_snd [simp]:
  fixes dsn' r
  shows \"\pi_6 (dsn', snd (r)) = \pi_6 (r)\"
  by (cases r) simp

3.1.3 Routing Tables
Routing tables map ip addresses to route entries.

type_synonym rt = \"ip \Rightarrow r\"

syntax
  \"_Sigma_route\" :: \"rt \Rightarrow ip \Rightarrow r\" (\"\sigma_{route}(\_, \_')\")

translations
  \"\sigma_{route}(rt, dip)\" \Rightarrow \"rt dip\"

definition sqn :: \"rt \Rightarrow ip \Rightarrow sqn\"
  where \"sqn rt dip \equiv case \sigma_{route}(rt, dip) of Some r \Rightarrow \pi_2 (r) \mid None \Rightarrow 0\"

definition sqnf :: \"rt \Rightarrow ip \Rightarrow k\"
  where \"sqnf rt dip \equiv case \sigma_{route}(rt, dip) of Some r \Rightarrow \pi_3 (r) \mid None \Rightarrow unk\"

abbreviation flag :: \"rt \Rightarrow ip \Rightarrow f\"
  where \"flag rt dip \equiv map_option \pi_4 (\sigma_{route}(rt, dip))\"

abbreviation dhops :: \"rt \Rightarrow ip \Rightarrow nat\"
  where \"dhops rt dip \equiv map_option \pi_5 (\sigma_{route}(rt, dip))\"

abbreviation nhop :: \"rt \Rightarrow ip \Rightarrow ip\"
  where \"nhop rt dip \equiv map_option \pi_6 (\sigma_{route}(rt, dip))\"

definition vD :: \"rt \Rightarrow ip set\"
  where \"vD rt \equiv \{dip. flag rt dip = Some val\}\"
definition \( \text{id} :: \text{"rt} \Rightarrow \text{ip set}\) where
\[ \text{id} \ rt \equiv \{\text{dip. flag rt dip} = \text{Some inv}\}\]

definition \( \text{kD} :: \text{"rt} \Rightarrow \text{ip set}\) where
\[ \text{kD} \ rt \equiv \{\text{dip. rt dip} \neq \text{None}\}\]

lemma \( \text{kD\_is\_vD\_and\_iD}: \text{"kD rt} = \text{vD rt} \cup \text{iD rt}\) unfolding \( \text{kD\_def vD\_def iD\_def}\) by auto

lemma \( \text{vD\_iD\_gives\_kD} \) [simp]:
\[ \forall \text{ip rt. ip} \in \text{vD rt} \implies \text{ip} \in \text{kd rt}\]
\[ \forall \text{ip rt. ip} \in \text{iD rt} \implies \text{ip} \in \text{kd rt}\]
unfolding \( \text{kD\_is\_vD\_and\_iD}\) by simp_all

lemma \( \text{kD\_Some} \) [dest]:
fixes \( \text{dip rt}\)
assumes "\( \text{dip} \in \text{kD rt}\)"
shows "\( \exists \text{dsn dsk flag hops nhip. } \sigma\text{route(rt, dip)} = \text{Some (dsn, dsk, flag, hops, nhip)}\)"
using assms unfolding \( \text{kD\_def}\) by simp

lemma \( \text{kD\_None} \) [dest]:
fixes \( \text{dip rt}\)
assumes "\( \text{dip} \notin \text{kD rt}\)"
shows "\( \sigma\text{route(rt, dip)} = \text{None}\)"
using assms unfolding \( \text{kD\_def}\) by (metis (mono_tags) mem_Collect_eq)

lemma \( \text{vD\_Some} \) [dest]:
fixes \( \text{dip rt}\)
assumes "\( \text{dip} \in \text{vD rt}\)"
shows "\( \exists \text{dsn dsk hops nhip. } \sigma\text{route(rt, dip)} = \text{Some (dsn, dsk, val, hops, nhip)}\)"
using assms unfolding \( \text{vD\_def}\) by simp

lemma \( \text{vD\_empty} \) [simp]: "\( \text{vD Map.empty} = \{}\)"
unfolding \( \text{vD\_def}\) by simp

lemma \( \text{iD\_Some} \) [dest]:
fixes \( \text{dip rt}\)
assumes "\( \text{dip} \in \text{iD rt}\)"
shows "\( \exists \text{dsn dsk hops nhip. } \sigma\text{route(rt, dip)} = \text{Some (dsn, dsk, inv, hops, nhip)}\)"
using assms unfolding \( \text{iD\_def}\) by simp

lemma \( \text{val\_is\_vD} \) [elim]:
fixes \( \text{ip rt}\)
assumes "\( \text{ip} \in \text{kD(rt)}\)"
and "\( \text{the (flag rt ip)} = \text{val}\)"
shows "\( \text{ip} \in \text{vD(rt)}\)"
using assms unfolding \( \text{vD\_def}\) by auto

lemma \( \text{inv\_is\_iD} \) [elim]:
fixes \( \text{ip rt}\)
assumes "\( \text{ip} \in \text{kD(rt)}\)"
and "\( \text{the (flag rt ip)} = \text{inv}\)"
shows "\( \text{ip} \in \text{iD(rt)}\)"
using assms unfolding \( \text{iD\_def}\) by auto

lemma \( \text{iD\_flag\_is\_inv} \) [elim, simp]:
fixes \( \text{ip rt}\)
assumes "\( \text{ip} \in \text{iD(rt)}\)"
shows "\( \text{the (flag rt ip)} = \text{inv}\)"
proof -
from \( \text{ip} \in iD(rt) \) have "ip \( \in kD(rt) " \ by auto
  with assms show ?thesis unfolding iD_def by auto
qed

lemma \( kD\text{-}but\text{-}not\_vD\text{-}is\_iD \) [elim]:
  fixes \( \text{ip} \) \( rt \)
  assumes "ip \( \in kD(rt) "
  and "ip \( \notin vD(rt) "
  shows "ip \( \in iD(rt) "
proof -
  from \( \text{ip} \in kD(rt) \) obtain dsn dsk f hops nhop
    where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop)"
    by (metis kD_Some)
  from \( \text{ip} \notin vD(rt) \) have "f \( \not= \) val"
proof (rule contrapos_nn)
    assume "f = val"
    with rtip have "the (flag rt ip) = val"
      by simp
    with \( \text{ip} \in kD(rt) \) show "ip \( \in vD(rt) " ..
  qed
  with rtip have "the (flag rt ip)= inv" by simp
  with \( \text{ip} \in kD(rt) \) show "ip \( \in iD(rt) " ..
qed

lemma \( vD\text{-}or\_iD \) [elim]:
  fixes \( \text{ip} \) \( rt \)
  assumes "ip \( \in kD(rt) "
  and "ip \( \in vD(rt) \) \( \ signifies \) \( \Rightarrow \) P rt ip"
  and "ip \( \in iD(rt) \) \( \ signifies \) \( \Rightarrow \) P rt ip"
  shows "P rt ip"
proof -
  from \( \text{ip} \in kD(rt) \) have "ip \( \in vD(rt) \cup iD(rt) \)"
    by (simp add: kD_is_vD_and_iD)
  thus ?thesis by (auto elim: assms(2-3))
qed

lemma proj5_eq_dhops: "\( \forall dip rt. dip \in kD(rt) \Rightarrow \pi_5 (the (rt dip)) = the (dhops rt dip) \)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj4_eq_flag: "\( \forall dip rt. dip \in kD(rt) \Rightarrow \pi_4 (the (rt dip)) = the (flag rt dip) \)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj2_eq_sqn: "\( \forall dip rt. dip \in kD(rt) \Rightarrow \pi_2 (the (rt dip)) = sqn rt dip \)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 [simp]:
  "\( \forall ip rt. ip \in kD(rt) \Rightarrow \text{sqnf rt ip } = \pi_3 (the (rt ip)) \)"
  unfolding sqnf_def by auto

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

lemma kD_update [simp]: "\( \forall rt nip v. kD (rt(nip \rightarrow v)) = insert nip (kD rt) \)"
  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' \lor ip \in kD(rt)"
  and "ip = ip' \Rightarrow P rt ip ip'"
  and "[ \text{ip } \not= \text{ip'; } ip \in kD(rt) ] \Rightarrow P rt ip ip'"
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

**Definition update_arg_wf :: "r ⇒ bool"**
where "update_arg_wf r r ≡ π₄(r) = val ∧ (∀nhip. update_arg_wf (0, unk, val, Suc 0, nhip)) ∧ (∀dsn dsk flag hops nhip. (dsn > 0 ⇒ P (dsn, kno, val, hops, nhip))"
and wf3: "dsk = unk \implies (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 \langle flag = val \rangle by simp (rule c1)

next
  assume "dsk = kno"
  moreover with wf2 have "dsn > 0" by simp
  ultimately show ?thesis using \langle flag = val \rangle 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) \land (\pi_5(s) > \pi_5(r) \lor \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
  "[\text{ip} \notin kD(rt)] \implies update rt ip r = rt (ip \mapsto r)"
  "[\text{ip} \in kD(rt); sqn rt ip < \pi_2(r)] \implies update rt ip r = rt (ip \mapsto r)"
  "[\text{ip} \in kD(rt); sqn rt ip = \pi_2(r);
  \text{the (dhops rt ip)} > \pi_5(r)] \implies update rt ip r = rt (ip \mapsto r)"
  "[\text{ip} \in kD(rt); sqn rt ip = \pi_2(r);
  flag rt ip = Some inv] \implies update rt ip r = rt (ip \mapsto r)"
  "[\text{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')"
  "[\text{ip} \in kD(rt); sqn rt ip \geq \pi_2(r); \pi_3(r) = kno;
  sqn rt ip = \pi_2(r) \implies (\text{the (dhops rt ip)} \leq \pi_5(r) \land \text{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 = π_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 = π_2(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 "π_3(r) = unk"
and "(π_2(r) = 0) = (π_3(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 (π_2(r) = 0) = (π_3(r) = unk) and (π_3(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 ≥ π_2(r)"
"π_3(r) = kno"
"sqn rt ip = π_2(r) ⇒ the (dhops rt ip) ≤ π_5(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 "(π_2(r) = 0) = (π_3(r) = unk)"
and c1: "[ [ip /∈ kD(rt) ] ] → P (rt (ip -> r))"

and c2: "[[ip ∈ kD(rt); sqn rt ip < π_2(r)]
⇒ P (rt (ip -> r))]
and c3: "[[ip ∈ kD(rt); sqn rt ip = π_2(r); the (dhops rt ip) > π_5(r)]
⇒ P (rt (ip -> r))]
and c4: "[[ip ∈ kD(rt); sqn rt ip = π_2(r); the (flag rt ip) = inv]
⇒ P (rt (ip -> r))]
and c5: "[[ip ∈ kD(rt); π_3(r) = unk]
⇒ P (rt (ip -> (π_3 (the σ_route(r, ip)), π_3(r),
π_4(r), π_5(r), π_6(r))))]
and c6: "[[ip ∈ kD(rt); sqn rt ip ≥ π_2(r); π_3(r) = kno;
sqn rt ip = π_2(r) ⇒ the (dhops rt ip) ≤ π_5(r) ∧ the (flag rt ip) = val]
⇒ P (rt (ip -> (the σ_route(r, 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 rt_eq: "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 \((\pi_2(r) = 0) = (\pi_3(r) = \text{unk})\);
c2 \[(\pi_2(r) < \pi_3(r) = \text{unk}) \Rightarrow P(rt(ip \mapsto r))\]
c3 \[\boxed{\text{sqn rt ip = } \pi_2(r); \text{ the (dhops rt ip) > } \pi_5(r) \Rightarrow P(rt(ip \mapsto r))}\]
c4 \[\boxed{\text{sqn rt ip = } \pi_2(r); \text{ the (flag rt ip) = inv} \Rightarrow P(rt(ip \mapsto r))}\]
c5 \[\boxed{\pi_3(r) = \text{unk} \Rightarrow P(rt(ip \mapsto r))}\]
c6 \[\boxed{\pi_3(r) = \text{unk} = \text{knob} \Rightarrow \text{sqn rt ip } \geq \pi_2(r); \pi_3(r) = \text{knob} \Rightarrow \text{the (dhops rt ip) } \leq \pi_5(r) \land \text{the (flag rt ip) = val} \Rightarrow P(rt(ip \mapsto \sigma_{\text{route}}(rt, ip)))}\]

shows "\(P(\text{update rt ip r})\)"
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 \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{knob}"
and "sqn rt ip = \pi_2(r) \Rightarrow \text{the (dhops rt ip) } \leq \pi_5(r) \land \text{the (flag rt ip) = val}"
thus "P(rt(ip \mapsto \sigma_{\text{route}}(rt, ip)))"
by (rule c6)

qed (simp add: \langle ip \in kD(rt) \rangle)

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 \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: "\forall v. rt dip = Some v \Rightarrow
update rt dip (dsn, dsk, flag, hops, nhip) \neq rt dip (dsn, dsk, flag, hops, nhip)"

260
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

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 flag hops nhip
  assumes "(dsn = 0) = (dsk = unk)"
  shows "sqn rt dip ≤ sqn (update rt rip (dsn, dsk, val, hops, nhip)) dip"
  proof (rule update_cases)
    show "(π₂ (dsn, dsk, val, hops, nhip) = 0) = (π₃ (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 ≤ hops"
  shows "sqn rt ip ≤ 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 "ip ≠ hop" (is: "∀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)) 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
  assumes "ip ≠ 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 ≠ 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)

261
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 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 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 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((invalidate rt dests) dip)) = π₃(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 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 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)"
  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
  assumes "invalidate rt dests dip = Some (dsn, dsk, flag, hops, nhip)"
  shows "dsn = (case dests dip of None ⇒ π 2 (the (rt dip)) | Some rsn ⇒ rsn)
  ∧ dsk = π 3 (the (rt dip))
  ∧ flag = (if dests dip = None then π 4 (the (rt dip)) else inv)
  ∧ hops = π 5 (the (rt dip))
  ∧ nhip = π 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) ⇒
  π 3 (the (invalidate rt dests dip)) = π 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)

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" ("σ_queue(_, _)"
  where "σ_queue(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 := None)
  | 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 \(\sigma_{p\text{-flag}}\) :: \(\text{store} \Rightarrow \text{ip} \rightarrow p\) \("\sigma_{p\text{-flag}}('_, _')"\)
where \(\sigma_{p\text{-flag}}(\text{store}, \text{dip}) \equiv \text{map_option} \text{fst} (\text{store dip})\)

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

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

3.1.7 Comparison with the original technical report

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

1. \text{nhop} is partial, thus a ‘the’ is needed, similarly for \text{dhops} and \text{addpreRT}.

2. \text{precs} 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

3.2 AODV protocol messages

theory \text{C\_Aodv\_Message}
imports \text{C\_Gtobcast}
begin

datatype \text{msg} =
  \text{Rreq} \text{ nat} \text{ rreqid} \text{ ip} \text{ sqn} \text{ k} \text{ ip} \text{ sqn} \text{ ip}
| \text{Rrep} \text{ nat} \text{ ip} \text{ sqn} \text{ ip} \text{ ip}
| \text{Rerr} "\text{ip} \Rightarrow \text{sqn}" \text{ ip}
| \text{Newpkt} \text{ data} \text{ ip}
| \text{Pkt} \text{ data} \text{ ip} \text{ ip}

instantiation \text{msg} :: \text{msg}
begin
  definition \text{newpkt\.def \[simp\]} :: "\text{newpkt} \equiv \lambda(d, dip). \text{Newpkt} d dip"
  definition \text{eq\_newpkt\.def} :: "\text{eq\_newpkt} m \equiv \text{case m of Newpkt d dip }\Rightarrow \text{ True} | \_ \Rightarrow \text{ False}"

  instance by intro_classes \(\text{simp add: eq\_newpkt\.def}\)
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 \text{rreq} :: \("\text{nat} \times \text{rreqid} \times \text{ip} \times \text{sqn} \times \text{k} \times \text{ip} \times \text{sqn} \times \text{ip} \Rightarrow \text{msg}\"
where \("\text{rreq} \equiv \lambda(hops, \text{rreqid}, \text{dip}, \text{dsn}, \text{dsk}, \text{oip}, \text{osn}, \text{sip}).
\text{Rreq} \text{ hops} \text{ rreqid} \text{ dip} \text{ dsn} \text{ dsk} \text{ oip} \text{ osn} \text{ sip}\"

lemma \text{rreq\_simp \[simp\]}:
"\text{rreq(hops, rreqid, dip, dsn, dsk, oip, osn, sip)} = \text{Rreq }\text{ hops} \text{ rreqid} \text{ dip} \text{ dsn} \text{ dsk} \text{ oip} \text{ osn} \text{ sip}\"
unfolding \text{rreq\.def} \text{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 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),
      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),
      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_notGlobals [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

3.3.2 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' ⇒
\{ ξ' \mid 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' \mid 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' \mid 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' \mid 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' \mid dests := dests', sip := sip' \}
_ \_ ⇒ \{\}

lemma is_rerr_asm [dest!]:
assumes "ξ' ∈ is_rerr ξ"
shows "(∃ dests' sip'.
msg ξ = Rerr dests' sip' ∧
ξ' = ξ' \mid 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 \implies ip ξ' = ip ξ"
"ξ' ∈ is_rrep ξ \implies ip ξ' = ip ξ"
"ξ' ∈ is_rreq ξ \implies ip ξ' = ip ξ"
"ξ' ∈ is_pkt ξ \implies ip ξ' = ip ξ"
"ξ' ∈ is_newpkt ξ \implies ip ξ' = ip ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp+) +

lemma is_msg_inv_sn [simp]:
"ξ' ∈ is_rerr ξ \implies sn ξ' = sn ξ"
"ξ' ∈ is_rrep ξ \implies sn ξ' = sn ξ"
"ξ' ∈ is_rreq ξ \implies sn ξ' = sn ξ"
"ξ' ∈ is_pkt ξ \implies sn ξ' = sn ξ"
"ξ' ∈ is_newpkt ξ \implies sn ξ' = sn ξ"
unfolding is_msg_defs
by (cases "msg ξ", clarsimp+) +

lemma is_msg_inv_rt [simp]:
"ξ' ∈ is_rerr ξ \implies rt ξ' = rt ξ"
\[ \xi' \in \text{is_rrep} \xi \implies \text{rt} \xi' = \text{rt} \xi \]
\[ \xi' \in \text{is_rreq} \xi \implies \text{rt} \xi' = \text{rt} \xi \]
\[ \xi' \in \text{is_pkt} \xi \implies \text{rt} \xi' = \text{rt} \xi \]
\[ \xi' \in \text{is_newpkt} \xi \implies \text{rt} \xi' = \text{rt} \xi \]

unfolding \text{is_msg_defs} by (cases "\text{msg} \xi", clarsimp+) +

lemma \text{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 \text{is_msg_defs} by (cases "\text{msg} \xi", clarsimp+) +

lemma \text{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 \text{is_msg_defs} by (cases "\text{msg} \xi", clarsimp+) +

lemma \text{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 \text{is_msg_defs} by (cases "\text{msg} \xi", clarsimp+) +

3.3.3 The protocol process

datatype \text{pseqp} =
\begin{itemize}
  \item PAodv
  \item PNewPkt
  \item PPkt
  \item PRreq
  \item PRrep
  \item PRerr
\end{itemize}

fun nat_of_seqp :: "\text{pseqp} \Rightarrow \text{nat}"
where
\begin{align*}
  \text{nat_of_seqp PAodv} & = 1 \\
  \text{nat_of_seqp PPkt} & = 2 \\
  \text{nat_of_seqp PNewPkt} & = 3 \\
  \text{nat_of_seqp PRreq} & = 4 \\
  \text{nat_of_seqp PRrep} & = 5 \\
  \text{nat_of_seqp PRerr} & = 6
\end{align*}

instantiation "\text{pseqp}" :: ord begin
  definition less_eq_seqp [iff]: "11 \leq 12 = (\text{nat_of_seqp} 11 \leq \text{nat_of_seqp} 12)"
  definition less_seqp [iff]: "11 < 12 = (\text{nat_of_seqp} 11 < \text{nat_of_seqp} 12)"

instance ..

end

abbreviation \text{AODV} where
\begin{align*}
  \text{AODV} & \equiv \lambda_. \text{[clear_locals] call(PAodv)}
\end{align*}

abbreviation \text{PKT} where
\begin{align*}
  \text{PKT} \text{ args} & \equiv
\end{align*}
\[ \begin{align*}
&\xi. \text{let \ (data, dip, oip) = args } \xi \text{ in} \\
&\quad (\text{clear_locals } \xi) \ (| \ \text{data := data, dip := dip, oip := oip}) \\
&\quad \text{call(PPkt)}
\end{align*} \]

abbreviation NEWPkt
where
\[
\text{NEWPkt \ args } \equiv \\
\xi. \text{let \ (data, dip) = args } \xi \text{ in} \\
\quad (\text{clear_locals } \xi) \ (| \ \text{data := data, dip := dip}) \\
\quad \text{call(PNewPkt)}
\]

abbreviation RREQ
where
\[
\text{RREQ \ args } \equiv \\
\xi. \text{let \ (hops, rreqid, dip, dsn, dsk, oip, osn, sip) = args } \xi \text{ in} \\
\quad (\text{clear_locals } \xi) \ (| \ \text{hops := hops, rreqid := rreqid, dip := dip,} \\
\quad \quad \text{dsn := dsn, dsk := dsk, oip := oip,} \\
\quad \quad \text{osn := osn, sip := sip}) \\
\quad \text{call(PRreq)}
\]

abbreviation RREP
where
\[
\text{RREP \ args } \equiv \\
\xi. \text{let \ (hops, dip, dsn, oip, sip) = args } \xi \text{ in} \\
\quad (\text{clear_locals } \xi) \ (| \ \text{hops := hops, dip := dip,} \\
\quad \text{dsn := dsn, oip := oip, sip := sip}) \\
\quad \text{call(PRrep)}
\]

abbreviation RERR
where
\[
\text{RERR \ args } \equiv \\
\xi. \text{let \ (dests, sip) = args } \xi \text{ in} \\
\quad (\text{clear_locals } \xi) \ (| \ \text{dests := dests, sip := sip}) \\
\quad \text{call(PRerr)}
\]

fun Γ \text{AODV :: } (\text{state, msg, pseqp, pseqp label}) \ \text{seqp_env}
where
\[
\begin{align*}
\Gamma \text{AODV} &= \text{PAodv \ (receive}(\lambda \text{msg' } \xi. \xi (| \ \text{msg := msg'})) \\
&\quad \circ \text{(is_newpkt) \ NEWPkt}(\lambda \xi. (\text{data } \xi, \text{ip } \xi)) \\
&\quad \circ \text{(is_pkt) \ PKT}(\lambda \xi. (\text{data } \xi, \text{dip } \xi, \text{oip } \xi)) \\
&\quad \circ \text{(is_rreq) \ RREQ}(\lambda \xi. (\text{hops } \xi, \text{rreqid } \xi, \text{dip } \xi, \text{dsn } \xi, \text{dsk } \xi, \text{oip } \xi, \text{osn } \xi, \text{sip } \xi)) \\
&\quad \circ \text{(is_rrep) \ RREP}(\lambda \xi. (\text{hops } \xi, \text{dip } \xi, \text{dsn } \xi, \text{oip } \xi, \text{sip } \xi)) \\
&\quad \circ \text{(is_rerr) \ RERR}(\lambda \xi. (\text{dests } \xi, \text{sip } \xi)) \\
&\quad \circ \text{unicast}(\lambda \xi. \{ \xi \ | \ \text{dip := dip} \}) \ | \ \text{dip, dip } \in \text{qD(store } \xi) \ \cap \text{vD}(\text{rt } \xi)) \\
&\quad \circ \text{broadcast}(\lambda \xi. \text{the } (\text{nhop } (\text{rt } \xi)) (\text{dip } \xi), \lambda \xi. \text{pkt(data } \xi, \text{dip } \xi, \text{ip } \xi)) \\
&\quad \circ \text{AODV}() \\
&\quad \circ \text{dests} := (\lambda \text{rip}. \text{if } (\text{rip } \in \text{vD}(\text{rt } \xi) \land \text{nhop } (\text{rt } \xi) \text{ rip } = \text{nhop } (\text{rt } \xi) (\text{dip } \xi)) \\
&\quad \quad \text{then Some (inc (sqn (rt } \xi) \text{ rip})) else None) \}) \\
&\quad \circ \text{rt} := \text{invalidate } (\text{rt } \xi) (\text{dests } \xi) \\
&\quad \circ \text{store} := \text{setRRF } (\text{store } \xi) (\text{dests } \xi) \\
&\quad \circ \text{unicast}(\lambda \xi. \text{the } (\text{nhop } (\text{rt } \xi)) (\text{dip } \xi), \lambda \xi. \text{pkt(data } \xi, \text{dip } \xi, \text{ip } \xi)) \\
&\quad \circ \text{broadcast}(\lambda \xi. \text{the } (\text{nhop } (\text{rt } \xi)) (\text{dip } \xi), \lambda \xi. \text{pkt(data } \xi, \text{dip } \xi, \text{ip } \xi)) \\
&\quad \circ \text{AODV}()
\end{align*}
\]
\[\Gamma_{\text{AODV}}\]

\[\text{PNewPkt} = \text{labelled PNewPkt}\ (\langle\xi.\ \text{dip}\ \xi = \text{ip}\ \xi\rangle.\ AODV())\]

\[\text{broadcast}(\lambda\xi.\ \text{rreq}(0,\ \text{rreqid}\ \xi,\ \text{dip}\ \xi,\ \text{sqn}\ (\text{rt}\ \xi)\ (\text{dip}\ \xi),\ \text{sqnf}\ (\text{rt}\ \xi)\ (\text{dip}\ \xi),\ \text{ip}\ \xi,\ \text{sn}\ \xi,\ \text{ip}\ \xi)).\ AODV())\]

\[\]
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. ξ () oip ξ = ip ξ) |
| AODV() |
| D (ξ. ξ () oip ξ ≠ ip ξ) |
| (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() |
| D (ξ. ξ () dests ξ ≠ Map.empty) |
| broadcast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(hops ξ + 1, dip ξ, |
  dsn ξ, oip ξ, ip ξ)). AODV() |
| ) |
| D (ξ. ξ () dests ξ ≠ Map.empty) |
| broadcast(λξ. rerr(dests ξ, ip ξ)). AODV() |

| "ΓAODV PRerr = labelled PRerr (ξ. rt ξ ≠ update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ)) |
| (ξ. ξ () dests := (λrip. case (dests ξ) rip of None ⇒ None |
  | Some rsn ⇒ 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() |
| D (ξ. ξ () dests ξ ≠ Map.empty) |
| broadcast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(hops ξ + 1, dip ξ, |
  dsn ξ, oip 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 PAodv = seqp_skeleton (ΓAODV PAodv)"
/ "ΓAODV_skeleton PNewPkt = seqp_skeleton (ΓAODV PNewPkt)"
/ "ΓAODV_skeleton PPkt = seqp_skeleton (ΓAODV PPkt)"
/ "ΓAODV_skeleton PReq = seqp_skeleton (ΓAODV PReq)"
/ "ΓAODV_skeleton PRep = seqp_skeleton (ΓAODV PRep)"
/ "ΓAODV_skeleton PErr = seqp_skeleton (ΓAODV PErr)"

lemma ΓAODV_skeleton_wf [simp]:
  "wellformed ΓAODV_skeleton"
proof (rule, intro allI)

273
fix \( p \) \( pn' \)
show \( \text{"call}(pn') \notin \text{stermsl} (\Gamma_{AODV\_skeleton}\ pn)" \)
by (cases \( pn \)) simp_all
qed

declare \( \Gamma_{AODV\_skeleton}\_simps \ [\text{simp del, code del}] \)
lemmas \( \Gamma_{AODV\_skeleton}\_simps \ [\text{simp, code}] \)
\hfill \( \text{= } \Gamma_{AODV\_skeleton}\_simps \ [\text{simplified } \Gamma_{AODV\_simps} \ \text{seqp}\_skeleton\_simps] \)

lemma \( \text{aodv}\_proc\_cases \ [\text{dest}] \): 
fixes \( p \ \text{pn} \)
shows \( \text{"} p \in \text{ctermsl} (\Gamma_{AODV} \ pn) \text{\"} \)
using \( \text{assms} \)
by (cases \( pn \)) simp_all

definition \( \sigma_{AODV} \) :: \( \text{ip} \Rightarrow \text{(state } \times \text{ (state, msg, pseqp, pseqp label) seqp) set} \)
where \( \text{"} \sigma_{AODV} i \equiv \{(\text{aodv}\_init i, \Gamma_{AODV} \ \text{PAodv})\} \text{\"} \)

abbreviation \( \text{paodv} \)
:: \( \text{ip} \Rightarrow \text{(state } \times \text{ (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton} \)
where \( \text{"} \text{paodv} i \equiv (\text{init } = \sigma_{AODV} i, \text{trans } = \text{seqp}\_sos \ \Gamma_{AODV} \ )\text{\"} \)

lemma \( \text{aodv}\_trans \): \( \text{"} \text{trans } (\text{paodv} i) = \text{seqp}\_sos \ \Gamma_{AODV} \text{\"} \)
by simp

lemma \( \text{aodv}\_control\_within \ [\text{simp}] \): \( \text{"} \text{control\_within } \Gamma_{AODV} (\text{init } (\text{paodv} i)) \text{\"} \)
unfolding \( \sigma_{AODV}\_def \) by (rule \( \text{control\_withinI} \)) (auto simp del: \( \Gamma_{AODV}\_simps \))

lemma \( \text{aodv}\_wf \ [\text{simpl}] \):
\( \text{"} \text{wellformed } \Gamma_{AODV} \text{\"} \)
proof (rule, intro allI)
fix \( pn \ pn' \)
show \( \text{"} \text{call}(pn') \notin \text{stermsl} (\Gamma_{AODV} \ pn) \text{\"} \)
by (cases \( pn \)) simp_all
qed

lemmas \( \text{aodv}\_labels\_not\_empty \ [\text{simpl}] = \text{labels\_not\_empty} \ [\text{OF aodv}\_wf] \)

lemma \( \text{aodv}\_ex\_label \ [\text{intro}] \):
\( \text{"} \exists l. l \in \text{labels} \ \Gamma_{AODV} \ p \text{\"} \)
by (metis aodv\_labels\_not\_empty all\_not\_in\_conv)

lemma \( \text{aodv}\_ex\_label\_E \ [\text{elim}] \):
assumes \( \text{"} \forall l \in \text{labels} \ \Gamma_{AODV} \ p. \ P \ l \ p \text{\"} \)
and \( \text{"} \exists p \ l . \ P \ l \ p \implies Q\text{\"} \)
shows \( \text{"} Q\text{\"} \)
using \( \text{assms} \) by (metis aodv\_ex\_label)

lemma \( \text{aodv}\_simple\_labels \ [\text{simpl}] \):
\( \text{"} \text{simple\_labels } \Gamma_{AODV} \text{\"} \)
proof
fix \( p \)
assume \( \text{"} p \in \text{subterms}(\Gamma_{AODV} \ pn) \text{\"} \)
thus \( \text{"} \exists l . \ \text{labels} \ \Gamma_{AODV} \ p = \{l\} \text{\"} \)
by (cases \( pn \)) (simp\_all cong: seqp\_congs | elim\_disjE)+
qed

lemma \( \sigma_{AODV}\_labels \ [\text{simpl}] \):
\( \text{"} (\xi, p) \in \sigma_{AODV} i \implies \text{labels } \Gamma_{AODV} \ p = \{\text{PAodv}\_0\} \text{\"} \)
unfolding \( \sigma_{AODV}\_def \) by simp
lemma aodv_init_kD_empty [simp]:
"(ξ, p) ∈ σ_AODV i ⇒ kD (rt ξ) = {}"
unfolding σ_AODV_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) ∈ σ_AODV i"
shows "sip ξ ≠ ip ξ"
using assms unfolding σ_AODV_def by simp

lemma aodv_init_sip_not_i [simp]:
assumes "(ξ, p) ∈ σ_AODV i"
shows "sip ξ ≠ i"
using assms unfolding σ_AODV_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 Γ_AODV_simps [cterms_env]
aodv_proc_cases [ctermsl_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

3.4 Invariant assumptions and properties

theory C_Aodv_Predicates
imports C_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 ⇒ ip"
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"
"⋀ d 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 _ _ _ _ _ _ _ _ oipc osn sipc ⇒ hopsc = 0 ⇒ oipc = sipc"
lemma msg_zhops_simps [simp]:

"\[ \text{msg_zhops} (\text{Rreq} \text{ hops} \text{ rreqid} \text{ dip} \text{ dsn} \text{ dsk} \text{ oip} \text{ osn} \text{ sip}) = (\text{hops} = 0 \rightarrow \text{oip} = \text{sip}) \]

"\[ \text{msg_zhops} (\text{Rrep} \text{ hops} \text{ dip} \text{ dsn} \text{ oip} \text{ sip}) = (\text{hops} = 0 \rightarrow \text{dip} = \text{sip}) \]

"\[ \text{msg_zhops} (\text{Rerr} \text{ dests} \text{ sip}) = \text{True} \]

unfolding msg_zhops_def by simp_all

definition rreq_rrep_sn :: "msg \Rightarrow bool"

where "rreq_rrep_sn m ≡ case m of
  Rreq _ _ _ _ _ _ oipc osnc _ ⇒ osnc \geq 1
  | Rrep _ _ dsnc _ _ ⇒ dsnc \geq 1
  | _ ⇒ True"

lemma rreq_rrep_sn_simps [simp]:

"\[ \text{rreq_rrep_sn} (\text{Rreq} \text{ hops} \text{ rreqid} \text{ dip} \text{ dsn} \text{ dsk} \text{ oip} \text{ osn} \text{ sip}) = (\text{osn} \geq 1) \]

"\[ \text{rreq_rrep_sn} (\text{Rrep} \text{ hops} \text{ dip} \text{ dsn} \text{ oip} \text{ sip}) = (\text{dsn} \geq 1) \]

"\[ \text{rreq_rrep_sn} (\text{Rerr} \text{ dests} \text{ sip}) = \text{True} \]

"\[ \text{rreq_rrep_sn} (\text{Newpkt} \text{ d} \text{ dip}) = \text{True} \]

"\[ \text{rreq_rrep_sn} (\text{Pkt} \text{ d} \text{ dip} \text{ sip}) = \text{True} \]

unfolding rreq_rrep_sn_def by simp_all

definition rreq_rrep_fresh :: "rt \Rightarrow msg \Rightarrow bool"

where "rreq_rrep_fresh crt m ≡ case m of
  Rreq hopsc _ _ _ _ _ _ oipc osnc ipcc _ ⇒ (ipcc \neq oipc \rightarrow
    \text{ipcc} \in kD(crt) \land (sqn crt oipc > osnc
    \lor (sqn crt oipc = osnc
    \land \text{the (dhops crt oipc)} \leq hopsc
    \land \text{the (flag crt oipc)} = \text{val})))
  | Rrep hopsc dipc dsnc _ ipcc _ ⇒ (ipcc \neq dipc \rightarrow
    \text{dipc} \in kD(crt)
    \land sqn crt dipc = dsnc
    \land \text{the (dhops crt dipc)} = hopsc
    \land \text{the (flag crt dipc)} = \text{val})
  | _ ⇒ True"

lemma rreq_rrep_fresh [simp]:

"\[ \text{rreq_rrep_fresh} (\text{Rreq} \text{ hops} \text{ rreqid} \text{ dip} \text{ dsn} \text{ dsk} \text{ oip} \text{ osn} \text{ sip}) =
  (\text{ipcc} \neq \text{oipc} \rightarrow
    \text{ipcc} \in kD(crt)
    \land (\text{sqn crt oipc} > \text{osnc}
    \lor (\text{sqn crt oipc} = \text{osnc}
    \land (\text{the (dhops crt oipc)} \leq \text{hopsc}
    \land (\text{the (flag crt oipc)} = \text{val})))
  )
  \land \text{rreq_rrep_fresh_def by simp_all}

definition rerr_invalid :: "rt \Rightarrow msg \Rightarrow bool"

where "rerr_invalid crt m ≡ case m of
  Rerr destsc _ ⇒ (\forall ripc \in \text{dom(destsc)}.
  \text{ripc} \in iD(crt) \land \text{the (destsc ripc)} = \text{sqn crt ripc})
  | _ ⇒ \text{True}"
rerr_invalid crt (Rreq hops rreqid dip dsn dsk oip osn sip) = True" 
"\\hops dip dsn oip sip. rerr_invalid crt (Rreq hops dsn dip dsn oip sip) = True" 
"\\dests sip. rerr_invalid crt (Rerr dests sip) = (\forall rip \in \text{dom(dests)}. 
\quad \text{rip} \in \text{iD}(crt) \land \text{the (dests rip)} = \text{sqn} \text{ 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 \Rightarrow \text{state option}) \times 'a \Rightarrow (\text{nat} \Rightarrow \text{state}) \times 'a"

where
"initmissing \sigma = (\lambda i. \text{case (fst \sigma) i of None \Rightarrow aodv_init i | Some s \Rightarrow s, snd \sigma})"

lemma not_in_net_ips_fst_init_missing [simp]:
assumes "i \notin \text{net_ips} \sigma"
shows "\text{fst (initmissing (netgmap \text{fst} \sigma)) i} = \text{aodv_init i}"
using assms unfolding initmissing_def by simp

lemma fst_initmissing_netmap_pair_fst [simp]:
"\text{fst (initmissing (netgmap (\lambda (p, q). (\text{fst (id p)}, \text{snd (id p)}, q)) s))} 
\quad = \text{fst (initmissing (netgmap \text{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_netmap_default_aodv_init_netlift:
"\text{fst (initmissing (netgmap \text{fst} s))} = \text{default aodv_init (netlift \text{fst} s)}"

unfolding initmissing_def default_def
by (simp add: fst_netmap_netlift del: One_nat_def)

definition
netglobal :: "((\text{nat} \Rightarrow \text{state}) \Rightarrow \text{bool}) \Rightarrow (\text{\text{state} \times 'b} \times 'c) \text{ net_state} \Rightarrow \text{bool}"

where
"\text{netglobal P} \equiv (\lambda s. P (\text{default aodv_init (netlift \text{fst} s)}))"

end

3.5 Quality relations between routes

theory C_Fresher
imports C_Aodv_Data
begin

3.5.1 Net sequence numbers

On individual routes

definition
nsqn_r :: "r \Rightarrow \text{sqn}"

where
"\text{nsqn}_r r \equiv \text{if } \pi_4(r) = \text{val} \lor \pi_2(r) = 0 \text{ then } \pi_2(r) \text{ else } (\pi_2(r) - 1)"

lemma nsqn_r_def':
"\text{nsqn}_r r = (\text{if } \pi_4(r) = \text{inv then } \pi_2(r) - 1 \text{ else } \pi_2(r))"

unfolding nsqn_r_def by simp

lemma nsqn_r_zero [simp]:
"\\dsn dsk flag hops nhip. nsqn_r (0, dsk, flag, hops, nhip) = 0"

unfolding nsqn_r_def byclarsimp

lemma nsqn_r_val [simp]:
"\\dsn dsk hops nhip. nsqn_r (dsn, dsk, val, hops, nhip) = dsn"

unfolding nsqn_r_def byclarsimp
lemma nsqn_r_inv [simp]:
"∀dsn dsk hops nhip. nsqn, (dsn, dsk, inv, hops, nhip) = dsn - 1"
unfolding nsqn_r_def by clarsimp

lemma nsqn_r_lte_dsn [simp]:
"∀dsn dsk flag hops nhip. nsqn, (dsn, dsk, flag, hops, nhip) ≤ dsn"
unfolding nsqn_r_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 ⇒ 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 (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
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:
"∀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

278
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_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 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 < nsqn, r') ∨ (nsqn, r = nsqn, r' ∧ π5(r) ≥ π5(r')))"

lemma fresherI1 [intro]:
  assumes "nsqn, r < nsqn, r'"
  shows "r ⊑ r'"
  unfolding fresher_def using assms by simp

lemma fresherI2 [intro]:
  assumes "nsqn, r = nsqn, r'"
  and "π5(r) ≥ π5(r')"
  shows "r ⊑ r'"
  unfolding fresher_def using assms by simp

lemma fresherI [intro]:
assumes "(nsqnr r < nsqnr r') \lor (nsqnr r = nsqnr r' \land \pi5(r) \geq \pi5(r'))"
        shows "r \sqsubseteq r'"
        unfolding fresher_def using assms .

lemma fresherE [elim]:
  assumes "r \sqsubseteq r'"
        and "nsqnr r < nsqnr r' \Longrightarrow P r r'"
        and "nsqnr r = nsqnr r' \land \pi5(r) \geq \pi5(r') \Longrightarrow 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 ] \Longrightarrow x \sqsubseteq z"
       unfolding fresher_def by auto

lemma not_fresher_trans [elim, trans]:
  "[ \neg(x \sqsubseteq y); \neg(z \sqsubseteq x) ] \Longrightarrow \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

3.5.3 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 ip rt2 \equiv rt_fresher ip rt1 rt2"

lemma rt_fresher_def':
  "((rt1 \sqsubseteq ip rt2) = (nsqnr (the (rt1 i))) < nsqnr (the (rt2 i)) \lor
   nsqnr (the (rt1 i)) = nsqnr (the (rt2 i)) \land \pi5 (the (rt2 i)) \leq \pi5 (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) = (nsqnr rt1 dip < nsqnr rt2 dip
               \lor (nsqnr rt1 dip = nsqnr rt2 dip
                  \land the (dhops rt1 dip) \geq the (dhops rt2 dip)))"
       using assms unfolding rt_fresher_def by (simp add: kD_nsqn proj5_eq_dhops)

lemma rt_fresherI1 [intro]:
  assumes "dip \in kD(rt1)"

and "\(\text{dip} \in kD(\text{rt2})\)"
and "\(\text{nsqn rt1 dip} < \text{nsqn rt2 dip}\)"
s
shows \("\text{rt1} \sqsubseteq_{\text{dip}} \text{rt2}\)"

unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt_fresherI2 [intro]:
assumes "\(\text{dip} \in kD(\text{rt1})\)"
and "\(\text{dip} \in kD(\text{rt2})\)"
and "\(\text{nsqn rt1 dip} = \text{nsqn rt2 dip}\)"
and "\(\text{the (dhops rt1 dip)} \geq \text{the (dhops rt2 dip)}\)"

shows \("\text{rt1} \sqsubseteq_{\text{dip}} \text{rt2}\)"

unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3) by simp

lemma rt_fresherE [elim]:
assumes "\(\text{rt1} \sqsubseteq_{\text{dip}} \text{rt2}\)"
and "\(\text{dip} \in kD(\text{rt1})\)"
and "\(\text{dip} \in kD(\text{rt2})\)"
and "\(\lfloor \text{nsqn rt1 dip} < \text{nsqn rt2 dip} \rfloor \implies \text{P rt1 rt2 dip}\)"
and "\(\lfloor \text{nsqn rt1 dip} = \text{nsqn rt2 dip}; \text{the (dhops rt1 dip)} \geq \text{the (dhops rt2 dip)} \rfloor \implies \text{P rt1 rt2 dip}\)"

shows "\(\text{P rt1 rt2 dip}\)"

using assms(1) unfolding rt_fresher_def2 [OF assms(2-3)]
using assms(4-5) by auto

lemma rt_fresher_refl [simp]: "\(\text{rt} \sqsubseteq_{\text{dip}} \text{rt}\)"

unfolding rt_fresher_def by simp

lemma rt_fresher_trans [elim, trans]:
assumes "\(\text{rt1} \sqsubseteq_{\text{dip}} \text{rt2}\)"
and "\(\text{rt2} \sqsubseteq_{\text{dip}} \text{rt3}\)"

shows "\(\text{rt1} \sqsubseteq_{\text{dip}} \text{rt3}\)"

using assms unfolding rt_fresher_def by auto

lemma rt_fresher_if_Some [intro!]:
assumes "\(\text{the (rt dip)} \sqsubseteq r\)"

shows "\(\text{rt} \sqsubseteq_{\text{dip}} (\lambda \text{ip}. \text{if ip = dip then Some r else rt ip})\)"

using assms unfolding rt_fresher_def by simp

definition rt_fresh_as :: \("\text{ip} \Rightarrow \text{rt} \Rightarrow \text{rt} \Rightarrow \text{bool}\"

where
"\(\text{rt_fresh_as} \equiv \lambda \text{dip rt1 rt2}. (\text{rt1} \sqsubseteq_{\text{dip}} \text{rt2}) \land (\text{rt2} \sqsubseteq_{\text{dip}} \text{rt1})\)"

abbreviation
\(\text{rt_fresh_as_syn} :: \("\text{rt} \Rightarrow \text{ip} \Rightarrow \text{rt} \Rightarrow \text{bool}\)\) "\(\langle/, \approx_\_\rangle\)" [51, 999, 51] 50

where
"\(\text{rt1} \approx_i \text{rt2} \equiv \text{rt_fresh_as i rt1 rt2}\)"

lemma rt_fresh_as_ref1 [simp]: "\(\\forall \text{dip. rt} \approx_{\text{dip}} \text{rt}\)"

unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_trans [simp, intro, trans]:
"\(\\forall \text{rt1 rt2 rt3 dip}. [ \text{rt1} \approx_{\text{dip}} \text{rt2}; \text{rt2} \approx_{\text{dip}} \text{rt3} ] \implies \text{rt1} \approx_{\text{dip}} \text{rt3}\)"

unfolding rt_fresh_as_def rt_fresher_def
by (metis (mono_tags) fresher_trans)

lemma rt_fresh_asI [intro!]:
assumes "\(\text{rt1} \sqsubseteq_{\text{dip}} \text{rt2}\)"
and "\(\text{rt2} \sqsubseteq_{\text{dip}} \text{rt1}\)"

shows "\(\text{rt1} \approx_{\text{dip}} \text{rt2}\)"

using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_fresherI [intro]:
assumes "\(\text{dip} \in kD(\text{rt1})\)"
and "\(\text{dip} \in kD(\text{rt2})\)"

shows "\(\text{rt1} \sqsubseteq_{\text{dip}} \text{rt2}\)"

using assms unfolding rt_fresher_def2 [OF assms(1-2)] using assms(3) by simp

281
and "the (rt1 dip) ⊑ the (rt2 dip)"
and "the (rt2 dip) ⊑ 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 \mathcal{D}(rt)"
and "dip \in \mathcal{D}(rt')"
and "\pi_5(\text{the (rt dip)}) = \pi_5(\text{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 \not\subseteq_{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' \not\subseteq_{dip} rt"
by (rule rt_fresherI2)

qed

lemma rt_fresh_asE [elim]:
assumes "rt1 \approx_{dip} rt2"
and "[ rt1 \not\subseteq_{dip} rt2; rt2 \not\subseteq_{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 \subseteq_{dip} rt2"

using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
assumes "rt1 \approx_{dip} rt2"
shows "rt2 \subseteq_{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 \subseteq_{dip} rt2)"
shows "\neg (rt1 \approx_{dip} rt2)"

proof
assume "rt1 \subseteq_{dip} rt2"
hence "rt1 \subseteq_{dip} rt2" ..
with \neg (rt1 \subseteq_{dip} rt2) show False ..

qed

lemma not_rt_fresh_asI2 [intro]:
assumes "\neg (rt2 \subseteq_{dip} rt1)"
shows "\neg (rt1 \approx_{dip} rt2)"

proof
assume "rt1 \approx_{dip} rt2"
hence "rt2 \subseteq_{dip} rt1" ..
with \neg (rt2 \subseteq_{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 "nsqn r (the (rt2 dip)) = nsqn r (the (rt1 dip))"
using assms(3) unfolding rt_fresh_as_def
by (auto simp: rt_fresher_def2 [OF ⟨dip ∈ kD(rt1)⟩ ⟨dip ∈ kD(rt2)⟩]
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 dsn_n dsk_n f_n hops_n nhip_n
where rttn [simp]: "the (rt dip) = (dsn_n, dsk_n, f_n, hops_n, nhip_n)"
by (metis prod_cases5)
with \(\langle \text{dhops rt dip} \rangle \geq 1\) and \(\langle \text{dip}\in\text{kD}(rt)\rangle\) have \(\text{hops}_n \geq 1\)

by (metis proj5_eq_dhops proj4_eq_dhops projs(4))

from \(\langle \text{dip}\in\text{kD}(rt)\rangle\); \(\text{rtn}\) have [simp]: \(\text{sqn} \text{ rt dip} = \text{dsn}_n\)

and [simp]: \(\langle \text{dhops} \text{ rt dip} \rangle = \text{hops}_n\)

and [simp]: \(\langle \text{flag} \text{ rt dip} \rangle = f_n\)

by (simp add: sqn_def proj5_eq_dhops [symmetric]
proj4_eq_flag [symmetric])+

from \(\langle \text{update_arg_wf r} \rangle\) have \((\text{dsn}_n, \text{dsk}_n, f_n, \text{hops}_n, \text{nhip}_n)\) ⊑ \(\langle \text{rtn} \rangle\)

by (simp add: sqn_def proj5_eq_dhops [symmetric])

hence \(\text{rt} \subseteq \text{dip} \langle \text{update rt dip r} \rangle\)

by (metis proj5_eq_dhops projs(4))

proof (rule wf_r_cases)

fix \(\text{nhip} \text{ pre}\)

from \(\langle \text{hops}_n \geq 1 \rangle\) have \(\langle \text{dsn}_n, \text{dsk}_n, f_n, \text{hops}_n, \text{nhip}_n\rangle\)

unfolding fresher_def sqn_def by (cases \(f_n\)) auto

thus \(\langle \text{dsn}_n, \text{dsk}_n, f_n, \text{hops}_n, \text{nhip}_n\rangle\)

unfolding fresher_def sqn_def by auto

next

fix \(\text{dsn} : : \text{sqn} \text{ and hops nhip pre}\)

assume \(0 < \text{dsn}\)

show \(\langle \text{dsn}_n, \text{dsk}_n, f_n, \text{hops}_n, \text{nhip}_n\rangle\)

unfolding fresher_def sqn_def by auto

proof (rule update_cases_kD [OF \(\langle \text{dip}\in\text{kD}(rt)\rangle\)], simp_all add: \(0 < \text{dsn}\))

assume \(\langle \text{dsn}_n < \text{dsn} \rangle\)

thus \(\langle \text{dsn}_n, \text{dsk}_n, f_n, \text{hops}_n, \text{nhip}_n\rangle\)

unfolding fresher_def sqn_def by auto

next

assume \(\langle \text{dsn}_n = \text{dsn} \rangle\)

and \(\langle \text{hops < hops}_n \rangle\)

thus \(\langle \text{dsn}, \text{dsk}_n, f_n, \text{hops}_n, \text{nhip}_n\rangle\)

unfolding fresher_def nsqn_r_def by simp

next

assume \(\langle \text{dsn}_n = \text{dsn} \rangle\)

with \(\langle 0 < \text{dsn} \rangle\)

show \(\langle \text{dsn}, \text{dsk}_n, \text{inv}, \text{hops}_n, \text{nhip}_n\rangle\)

unfolding fresher_def sqn_def by simp

qed

qed

hence \(\text{rt} \subseteq \text{dip} \langle \text{update rt dip r} \rangle\)

by (rule single_rt_fresher, simp)

with \(\langle \text{dip} = \text{ip} \rangle\) show \(?\text{thesis}\) by simp

qed

theorem rt_fresher_invalidate [simp]:
assumes \(\langle \text{dip}\in\text{kD}(rt)\rangle\)

and \(\langle \text{indestds} : : \forall \text{rip}\in\text{dom}(\text{dests}). \text{rip}\in\text{vD}(rt) \land \text{sqn} \text{ rt rip} < \text{the (dests rip)}\rangle\)

shows \(\langle \text{rt} \subseteq \text{dip} \text{ invalidate rt dests}\rangle\)

proof (cases \(\langle \text{dip}\in\text{dom}(\text{dests})\rangle\))

assume \(\langle \text{dip}\in\text{dom}(\text{dests})\rangle\)

thus \(?\text{thesis}\) using \(\langle \text{dip}\in\text{kD}(rt)\rangle\)

by (rule single_rt_fresher, simp)

next

assume \(\langle \text{dip}\in\text{dom}(\text{dests})\rangle\)

moreover with \(\langle \text{indestds} \rangle\) have \(\langle \text{dip}\in\text{vD}(rt)\rangle\)

and \(\langle \text{sqn} \text{ rt dip} < \text{the (dests dip)}\rangle\)

by auto

ultimately show \(?\text{thesis}\)

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 "nsqn_r (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 "sqn rt dip = nsqn_r (the (invalidate rt dests dip))" by auto
from ⟨dip∈kD(rt)⟩ have "nsqn_r (the (invalidate rt dests dip)) = the (dests dip) - 1"
using ⟨dip∈vD(rt)⟩ unfolding invalidate_def
with ⟨dip∈kD(rt)⟩ have "sqn rt dip = nsqn_r (the (invalidate rt dests dip))" by simp
qed
also have "nsqn_r (the (invalidate rt dests dip)) = nsqn (invalidate rt dests dip)" by simp
finally show "nsqn rt dip = nsqn (invalidate rt dests dip)".
qed simp

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) ∧ ¬(rt2 ⊑ dip rt1)"

abbreviation
rt_strictly_fresher_syn :: "rt ⇒ ip ⇒ rt ⇒ bool" ("_<|_" _ "_" ") [51, 999, 51] 50
where
"rt1 <| rt2 ≡ rt_strictly_fresher i rt1 rt2"

lemma rt_strictly_fresher_def'':
"rt1 <| rt2 = ((rt1 <| dip rt2) ∧ ¬(rt2 <| dip rt1))"
unfolding rt_strictly_fresher_def rt_fresh_as_def by auto

lemma rt_strictly_fresherI' [intro]:
assumes "rt1 <| rt2"
and "¬(rt2 <| rt1)"
shows "rt1 <| rt2"
using assms unfolding rt_strictly_fresher_def'' by simp

lemma rt_strictly_fresherE' [elim]:

assumes "rt1 ⊏ rt2"
  and "[ rt1 ⊏ rt2; ¬(rt2 ⊏ 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 ⊏ rt2"
  and "¬(rt1 ≈ rt2)"
  shows "rt1 ⊏ 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 ⊏ rt2"
  and "[ rt1 ⊏ rt2; ¬(rt1 ≈ 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 ⊏ 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 ⊏ dip rt2"
  shows "the (rt1 dip) ⊑ 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 ⊏ dip rt2"
  shows "¬ rt1 ≈ dip rt2"
using assms unfolding rt_strictly_fresher_def rt_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
  assumes "rt1 ⊏ dip rt2"
  and "rt2 ⊏ dip rt3"
  shows "rt1 ⊏ dip rt3"
using assms proof -
  from rt1 ⊏ dip rt2: obtain "the (rt1 dip) ⊑ the (rt2 dip)" by auto
  also from rt2 ⊏ dip rt3: obtain "the (rt2 dip) ⊑ the (rt3 dip)" by auto
  finally have "the (rt1 dip) ⊑ the (rt3 dip)" .
  moreover have "¬ (rt1 ≈ dip rt3)"
  proof -
    from rt1 ⊏ dip rt2: obtain "¬(the (rt2 dip) ⊑ the (rt1 dip))" by auto
    also from rt2 ⊏ dip rt3: obtain "¬(the (rt3 dip) ⊑ the (rt2 dip))" by auto
    finally have "¬(the (rt3 dip) ⊑ the (rt1 dip))" .
    thus thesis ..
  qed
  ultimately show "rt1 ⊏ dip rt3" ..
  qed

lemma rt_strictly_fresher_irefl [simp]: "¬ (rt ⊏ dip rt)"
using assms unfolding rt_strictly_fresher_def
by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
  assumes "rt1 ⊏ dip rt2"
  and "rt2 ⊏ dip rt3"
  shows "rt1 ⊏ 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 `¬ (rt3 ⊑ dip rt2)` 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 \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 rt2"

using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

lemma invalidate_rtstr_left [simp]:
"\\forall \text{dests dip rt rt'. dests dip = None } \implies \text{(invalidate rt dests } \sqsubseteq \text{ dip rt')} = \text{ (rt } \sqsubseteq \text{ 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 \text{ dip rt2}) = (rt1 \sqsubseteq \text{ dip rt2})"
proof
(cases "dip \in \text{dom(dests)}")

assume "dip \in \text{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 \text{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!]:
"\\forall dip ip rt r rt'. \ [ \ [ dip \neq ip; rt \sqsubseteq \text{ dip rt'}] \] = \implies update rt ip r \sqsubseteq \text{ 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) \sqsubseteq 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 \in vD(rt2 nhip)⟩ have "nsqn, (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 = nsqn, (the (rt2 nhip dip))"

unfolding nsqn_def using ⟨dip \in vD (rt2 nhip)⟩
by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
finally show "nsqn, (the (update rt dip (osn, kno, val, hops, nhip) dip)) < nsqn, (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) \sqsubseteq 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 "nsqn_r (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 = nsqn_r (the (rt2 nhip dip))"
  unfolding nsqn_r_def using (dip∈vD(rt2 nhip))
    by (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
finally show "nsqn_r (the (update rt dip (osn, kno, val, Suc hops, nhip) dip))
  = nsqn_r (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)"
  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)"
sows "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

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 Γ AODV \((λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')\)"
by inv_cterms

lemma sequence_number_one_or_bigger:
  "paodv i |||= onl Γ AODV \((λ(ξ, _). 1 ≤ sn ξ)\)"
by (rule onll_step_to_invariantI [OF sequence_number_increases])
  (auto simp: σ_AODV_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 Γ_AODV (λ(ξ, l), 1) \in \{PAodv-:7\} \cup \{PAodv-: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: 
"paodv i |= onl Γ_AODV (λ(ξ, 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: 
"paodv i |= onl Γ_AODV (λ(ξ, 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' = ξ[r : 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 
  \lor 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' = ξ[r : update (rt \xi) ip (sn, kno, val, Suc hops, sip)]" 
  and "sip \in kD (rt \xi)" 
  hence "\forall dip \in kD (rt \xi).
  the (nhop (update (rt \xi) ip (sn, kno, val, Suc hops, sip)) dip) = ip 
  \lor 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 aodv_wf sip_in_kD] solve: one_hop nhip_is_sip)
qed

Proposition 7.4

lemma known_destinations_increase: 
"paodv i |=A onll Γ_AODV (λ((ξ, _), _, (ξ', _)). kD (rt \xi) \subseteq kD (rt \xi'))"
by (inv_cterms simp add: subset_insertI)

Proposition 7.5

lemma rreqs_increase: 
"paodv i |=A onll Γ_AODV (λ((ξ, _), _, (ξ', _)). rreqs ξ \subseteq rreqs ξ')"
by (inv_cterms simp add: subset_insertI)

lemma dests_bigger_than_sqn: 
"paodv i |= onl Γ_AODV (λ(ξ, l). l \in \{PAodv-:15..PAodv-: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 ξ). ip \in kD(rt \xi) \land sqn (rt \xi) ip \leq the (dests ξ ip)))"
proof -
  have sqninv: 
    "\forall dests rt rsn ip. 
    [ \forall ip \in dom(dests). ip \in kD(rt) \land sqn rt ip \leq the (dests ip); dests ip = Some rsn ]
    \rightarrow sqn (invalidate rt dests) ip \leq rsn"
  by (rule sqn_invalidate_in_dests [THEN eq_imp_le], assumption) auto
  have indests:
Proof -
\begin{enumerate}
\item \textbf{fix} \(\xi : \text{state}\)
\item \textbf{assume} \(\ast: \forall \text{ip} \in \text{dom}(\text{dests} \xi). \text{ip} \in kD (\text{rt} \xi) \land \text{sqn (rt} \xi) \text{ip} \leq \text{the (dests} \xi \text{ip})\)
\item \textbf{have} \(\forall \text{ip}. \text{sqn (rt} \xi) \text{ip} \leq \text{sqn (invalidate (rt} \xi) \text{(dests} \xi)) \text{ip}\)
\item \textbf{proof} \textbf{fix} \text{ip} \textbf{from} \ast \textbf{have} \(\text{ip} \not\in \text{dom}(\text{dests} \xi) \lor \text{sqn (rt} \xi) \text{ip} \leq \text{the (dests} \xi \text{ip})\) \textbf{by} \text{simp} \textbf{thus} \(\text{sqn (rt} \xi) \text{ip} \leq \text{sqn (invalidate (rt} \xi) \text{(dests} \xi)) \text{ip}\) \textbf{by} \text{metis domI invalidate_sqn option.sel}
\end{enumerate}
\textbf{note} solve_invalidate = this
\textbf{show} \(\text{?thesis}\) \textbf{by} \text{inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_bigger_than_sqn] simp add: solve_invalidate}
\textbf{qed}

\begin{proof}
\begin{enumerate}
\item \textbf{lemma} sqns_increase:
\begin{itemize}
\item \textbf{"paodv i \(\models\) onl \(\Gamma_AODV\ (\lambda((\xi, _), _). \text{ip} \xi = \text{i})\"
\item \textbf{by} \text{inv_cterms simp add: \(\sigma_{AODV}\_\text{def}\)}
\end{itemize}
\item \textbf{lemma} sender_ip_valid':
\begin{itemize}
\item \textbf{"paodv i \(\models\) onl \(\Gamma_AODV\ (\lambda((\xi, _), a, _). \text{anycast (\(\lambda\text{m. not_Pkt m} \rightarrow \text{msg_sender m = ip} \xi\text{a})\})\"
\item \textbf{by} \text{inv_cterms}
\end{itemize}
\item \textbf{lemma} sender_ip_valid:
\begin{itemize}
\item \textbf{"paodv i \(\models\) onl \(\Gamma_AODV\ (\lambda((\xi, _), a, _). \text{anycast (\(\lambda\text{m. not_Pkt m} \rightarrow \text{msg_sender m = i}\text{a})\})\"
\item \textbf{by} \text{rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid']}
\end{itemize}
\item \textbf{lemma} received_msg_inv:
\begin{itemize}
\item \textbf{"paodv i \(\models\) (recvmsg P \rightarrow) onl \(\Gamma_AODV\ (\lambda(\xi, 1). \text{l} \in \{PAodv--:1\} \rightarrow P \text{(msg} \xi))\"
\item \textbf{by} \text{inv_cterms}
\end{itemize}
\item \textbf{lemma} sip_not_ip':
\begin{itemize}
\item \textbf{"paodv i \(\models\) (recvmsg (\(\lambda\text{m. not_Pkt m} \rightarrow \text{msg_sender m} \neq \text{i}\) \rightarrow) onl \(\Gamma_AODV\ (\lambda(\xi, _). \text{sip} \xi \neq \text{ip} \xi))\"
\item \textbf{by} \text{inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv] onl_invariant_sterms [OF aodv_wf ip_constant [THEN invariant_restrict_inD]] simp add: clear_locals_sip_not_ip') clarsimp+}
\end{itemize}
\item \textbf{lemma} sip_not_ip:
\begin{itemize}
\item \textbf{"paodv i \(\models\) (recvmsg (\(\lambda\text{m. not_Pkt m} \rightarrow \text{msg_sender m} \neq \text{i}\) \rightarrow) onl \(\Gamma_AODV\ (\lambda(\xi, _). \text{sip} \xi \neq \text{i})\"
\item \textbf{by} \text{inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv] onl_invariant_sterms [OF aodv_wf ip_constant [THEN invariant_restrict_inD]] simp add: clear_locals_sip_not_ip') clarsimp+}
\end{itemize}
\end{enumerate}
\textbf{Neither} sip_not_ip' \textbf{nor} sip_not_ip \textbf{is needed to show loop freedom.}

Proposition 7.10
lemma hop_count_positive:
"paodv i |= onl \( \Gamma_{AODV} \) (\( \lambda(\xi, \_). \forall ip \in kD (rt \xi). \text{the}(dhops (rt \xi) ip) \geq 1)\)"
by (inv_cterms) auto

lemma rreq_dip_in_vD_dip_eq_ip:
"paodv i |= onl \( \Gamma_{AODV} \) (\( \lambda(\xi, \_). \forall ip \in kD (rt \xi)). \text{the}(dhops (rt \xi) ip) \geq 1)\)"
by (inv_cterms) auto

Proposition 7.11
lemma anycast_msg_zhops:
"\( \forall rreqid dip dsn dsk oip osn sip. \)
paodv i \( \|\| \) \( \Gamma_{AODV} \) (\( \lambda(_, a, \_). \text{anycast msg_zhops a} \)\)
proof (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_dip_in_vD_dip_eq_ip [THEN invariant_restrict_inD]])
fix \( l \xi a pp p' pp' \)
assume "\( (\xi, pp) \in \text{reachable}(paodv i) TT \)"
and \( \{\text{PRreq-:14}\} \text{unicast}(\lambda \xi. \text{the}(nhop (rt \xi) (oip \xi)), \lambda \xi. \text{Rrep}(\text{the}(dhops (rt \xi) (dip \xi))) (dip \xi) (sqn (rt \xi) (dip \xi)) (oip \xi) (ip \xi)). \)
p' \( \triangleq \) pp' \( \in \text{sterms} \) \( \Gamma_{AODV} pp \)"
and \( l = \text{PRreq-:14} \) and \( a = \text{unicast}(\text{the}(nhop (rt \xi) (oip \xi))) (\text{Rrep}(\text{the}(dhops (rt \xi) (dip \xi))) (dip \xi) (sqn (rt \xi) (dip \xi)) (oip \xi) (ip \xi)) \) and 
*: \( \forall ip \in kD (rt \xi). \text{Suc 0} \leq \text{the}(dhops (rt \xi) ip) \) and
\( \text{dip } \xi \in vD (rt \xi) \) have "\( \text{dip } \xi \in kD (rt \xi) \)"
by (rule vD_iD_gives_kD(1))
with * have "\( \text{Suc 0} \leq \text{the}(dhops (rt \xi) (dip \xi)) \) ..
thus "\( 0 < \text{the}(dhops (rt \xi) (dip \xi)) \)" by simp
qed

lemma hop_count_zero_oip_dip_sip:
"paodv i \( \|\| \) \( \lambda(\xi, \_ a, \_ a) \). \text{anycast msg_zhops a} \)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) auto

lemma osn_rreq:
"paodv i \( \|\| \) \( \lambda(\xi, 1). \) \( \{\text{PAodv-:4..PAodv-:5} \cup \{\text{PRreq-:n|n. True}\} \} \rightarrow \text{hops}(\xi = 0 \rightarrow oip(\xi = \text{sip}(\xi))) \)\)
\( \land \)
\( \{\{\text{PAodv-:6..PAodv-:7} \cup \{\text{PRrep-:n|n. True}\} \} \rightarrow \text{hops}(\xi = 0 \rightarrow \text{dip}(\xi = \text{sip}(\xi)))\} \)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma osn_rreq':
"paodv i \( \|\| \) \( \lambda(\text{m. req}\_\text{rrep}\_\text{sn m} \land \text{msg_zhops m}) \) a""\text{recmsg}(\lambda(\text{m. req}\_\text{rrep}\_\text{sn m} \land \text{msg_zhops m}) \) a"
thus "\text{recmsg}(\lambda m. \text{rreq}\_\text{rrep}\_\text{sn m}) a"
by (cases a) simp_all
qed

lemma dsn_rrep:
"paodv i \( \|\| \) \( \lambda(\text{m. req}\_\text{rrep}\_\text{sn m} \land \text{msg_zhops m}) \) a"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma dsn_rrep':

292
\texttt{proof (rule invariant_weakenE [OF dsn_rrep])}

\texttt{fix a}

\texttt{assume \textit{"recvmsg (\lambda m. rreq_rrep_sn m \land msg_zhops m) a"}}

\texttt{thus \textit{"recvmsg rreq_rrep_sn a"}}

\texttt{by (cases a) simp_all}

qed

\textbf{Lemma hop_count_zero_oip_dip_sip':}

\texttt{proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])}

\texttt{fix a}

\texttt{assume \textit{"recvmsg (\lambda m. rreq_rrep_sn m \land msg_zhops m) a"}}

\texttt{thus \textit{"recvmsg msg_zhops a"}}

\texttt{by (cases a) simp_all}

qed

\textbf{Proposition 7.12}

\textbf{Lemma zero_seq_unk_hops_one':}

\texttt{proof -}

\{ \texttt{fix dip and \(\xi\) :: state and P}

\texttt{assume "sqn (inval (\rt (\xi)) (dests (\xi))) dip = 0"}

\texttt{and all: "\forall ip. sqn (\rt (\xi)) ip \leq sqn (inval (\rt (\xi)) (dests (\xi))) ip"}

\texttt{and *: "sqn (\rt (\xi)) dip = 0 \implies P (\xi dip)"}

\texttt{have "P (\xi dip)"}

\texttt{proof -}

\texttt{from all have "sqn (\rt (\xi)) dip \leq sqn (inval (\rt (\xi)) (dests (\xi))) dip" ..}

\texttt{with \texttt{sqn (\rt (\xi)) (dests (\xi))) dip = 0} \texttt{have "sqn (\rt (\xi)) dip = 0" by simp}

\texttt{thus "P (\xi dip)" by (rule *)}

\texttt{qed}

\texttt{note sqn_invalidate_zero [elim!] = this}

\{ \texttt{fix dip and \(\xi\) :: state and P}

\texttt{assume "\forall dip \in kD (\rt). (sqn (\rt) dip = 0 \implies sqn (\rt dip) dip = unk) \\
\land (sqn (\rt) dip = unk \implies dhops (\rt dip) dip = Suc 0) \\
\land (dhops (\rt dip) dip = Suc 0 \implies nhop (\rt dip) dip = dip)"

\texttt{proof -}

\texttt{from all have "sqn (\rt (\xi)) dip \leq sqn (inval (\rt (\xi)) (dests (\xi))) dip" ..}

\texttt{with \texttt{sqn (\rt (\xi)) (dests (\xi))) dip = 0} \texttt{have "sqn (\rt (\xi)) dip = 0" by simp}

\texttt{thus "P (\xi dip)" by (rule *)}

\texttt{qed}

\texttt{note prreq_ok1 [simp] = this}

\{ \texttt{fix dip and \(\xi\) :: state and P}

\texttt{assume "\forall dip \in kD (\rt). (sqn (\rt dip) dip = 0 \implies \pi_3 (\rt dip) dip = unk) \\
\land (\pi_3 (\rt dip) dip = unk \implies dhops (\rt dip) dip = Suc 0) \\
\land (dhops (\rt dip) dip = Suc 0 \implies nhop (\rt dip) dip = dip)"

\texttt{and "hops = 0 \implies dip = sip"}

\texttt{and "Suc 0 \leq dip"}

\texttt{and "ip \neq dip \implies dip \in kD (\rt)"

\texttt{hence "\texttt{the (dhops (update rt dip (dsn, kno, val, Suc hops, sip)) dip) = Suc 0} \implies \\
\texttt{the (nhop (update rt dip (dsn, kno, val, Suc hops, sip)) dip) = dip}"

\texttt{by - (rule update_cases, auto simp add: sqn_def dest!: bepec)}

\texttt{note prreq_ok1 [simp] = this}

\{ \texttt{fix dip and \(\xi\) :: state and P}

\texttt{assume "\forall dip \in kD (\rt). (sqn (\rt dip) dip = 0 \implies \pi_3 (\rt dip) dip = unk) \\
\land (\pi_3 (\rt dip) dip = unk \implies dhops (\rt dip) dip = Suc 0) \\
\land (dhops (\rt dip) dip = Suc 0 \implies nhop (\rt dip) dip = dip)"

\texttt{and "Suc 0 \leq dip"}

\texttt{and "ip \neq dip \implies dip \in kD (\rt)"

\texttt{hence "\pi_3 (\texttt{the (update rt dip (dsn, kno, val, Suc hops, sip)) dip}) = unk \implies \\
\texttt{the (dhops (update rt dip (dsn, kno, val, Suc hops, sip)) dip}) = Suc 0"}
by - (rule update_cases, auto simp add: sqnf_def sqn_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) ∧
    (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 aodv_wf hop_count_zero_oip_dip_sip']
   seq_step_invariant_sterms TT [OF sqns_increase aodv_wf aodv_trans]
   onl_invariant_sterms [OF aodv_wf osn_rreq']
   onl_invariant_sterms [OF aodv_wf dsn_rrep']) clarsimp+
qed

lemma zero_seq_unk_hops_one:
  "paodv i ||= (recvmsg (λ m. rreq_rrep_sn m ∧ msg_zhops m) → onl ΓAODV (λ (ξ, _).
    ∀ dip ∈ kD(rt ξ). (sqn (rt ξ) dip = 0 → (sqn (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:
  "paodv i ||= (recvmsg rreq_rrep_sn → onl ΓAODV (λ (ξ, 1).
    ∀ dip ∈ kD(rt ξ). π₃ (the (rt ξ dip)) = unk ∨ 1 ≤ π₂ (the (rt ξ dip)))"
proof -
  { fix sip rt dns1 dns2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2
    assume "dsk₁ = unk ∨ Suc 0 ≤ dns2"
    hence "π₃ (the (update rt sip (dns1, dsk1, flag1, hops1, nhip1) sip)) = unk ∨
      Suc 0 ≤ sqn (update rt sip (dns2, 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
} 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 [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 (recvmsg rreq_rrep_sn ↦ onll Γ\textsubscript{\textsc{aodv}} (λ(_ , a , _). anycast rreq_rrep_sn a)"

295
proof -

have sqnf_kno: "paodv i ⪯ onl \Gamma_{AODV} (\lambda(\xi, 1).
(1 \in \{PRreq-:14\} \rightarrow dip \ \xi \in kD (rt \ \xi) \land sqn (rt \ \xi) (dip \ \xi) = 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 ⪯ onll \Gamma_{AODV} (\lambda((\xi, _), a, _). anycast (rreq_rrep_fresh (rt \ \xi)) a)"
proof -

have rreq_oip: "paodv i ⪯ onl \Gamma_{AODV} (\lambda((\xi, l). (l \in \{PRreq-:3, PRreq-:4, PRreq-:13, PRreq-:21} 
\rightarrow oip \ \xi \in kD(rt \ \xi) 
\land (sqn (rt \ \xi) (oip \ \xi) > (osn \ \xi) 
\lor (sqn (rt \ \xi) (oip \ \xi) = (osn \ \xi) 
\land the (dhops (rt \ \xi) (oip \ \xi)) \leq Suc (hops \ \xi) 
\land the (flag (rt \ \xi) (oip \ \xi)) = val))))" by (inv_cterms inv add: onl_invariant_sterms [ OF aodv_wf rrep_1_update_changes]
onl_invariant_sterms [ OF aodv_wf sip_in_kD])
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 ⪯ onll \Gamma_{AODV} (\lambda((\xi, _), a, _). anycast (rerr_invalid (rt \ \xi)) a)"
proof -

have rrep_prrep: "paodv i ⪯ onl \Gamma_{AODV} (\lambda((\xi, l'). pp p')法律规定 asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts asserts assert
\[
\forall ip \in \text{dom}(\text{dests} \xi). \ ip \in \text{vD}(\text{rt} \xi)) \\
\land (1 \in \{\text{PAodv:-16..PAodv:-17}\} \cup \{\text{PPkt:-8..PPkt:-9}\} \cup \{\text{PRreq:-10..PRreq:-11}\} \cup \{\text{PRreq:-18..PRreq:-19}\} \cup \{\text{PRrep:-9..PRrep:-10}\} \cup \{\text{PRerr:-2..PRerr:-4}\} \rightarrow (\forall ip \in \text{dom}(\text{dests} \xi). \ ip \in \text{vD}(\text{rt} \xi)) \\
\land \text{the}(\text{dests} \xi ip) = \text{sqn}(\text{rt} \xi ip)) \\
\land (l = \text{PPkt:-12} \rightarrow dip \xi \in \text{iD}(\text{rt} \xi))"
\]

by inv_cterms (clarsimp split: split_if_asm option.split_asm simp: domIff)+

show ?thesis by (inv_cterms inv add: onl_invariant_sterms [OF aodv_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 \(\text{dests} of \text{type} ip \rightarrow \text{sqn}\), or to the first argument of the function \(\text{rerr}\), 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:

\[
\text{paodv } i \mid= \\
onl \Gamma_{\text{AODV}} (\lambda(\xi, \_). (1 \in \{\text{PAodv:-15, PPkt:-7, PRreq:-9, PRreq:-17, PRrep:-8}\} \\
\rightarrow (\forall ip \in \text{dom}(\text{dests} \xi). \ ip \in \text{vD}(\text{rt} \xi) \land \text{the}(\text{dests} \xi ip) = \text{inc}(\text{sqn}(\text{rt} \xi ip))) \\
\land (l = \text{PRerr:-1} \rightarrow (\forall ip \in \text{dom}(\text{dests} \xi). \ ip \in \text{vD}(\text{rt} \xi) \land \text{the}(\text{dests} \xi ip) > \text{sqn}(\text{rt} \xi ip))))
\]

by inv_cterms (clarsimp split: split_if_asm option.split_asm)+

Proposition 7.27

lemma route_tables_fresher:

\[
\text{paodv } i \mid= \_ \rightarrow \text{recvmsg } rreq_rrep_sn \rightarrow \text{onnll } \Gamma_{\text{AODV}} (\lambda((\xi, \_), \_). (\xi', \_)). \\
\forall dip \in \text{kD}(\text{rt} \xi). \text{rt} \xi \subseteq dip \text{rt} \xi'
\]

proof (inv_cterms inv add:

onl_invariant_sterms [OF aodv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aodv_wf osn_rreq]
onl_invariant_sterms [OF aodv_wf dsn_rrep]
onl_invariant_sterms [OF aodv_wf invariant_restrict_inD])

fix \xi \ pp \ p'\n
assume "((\xi, pp) \in \text{reachable } (\text{paodv } i ) \ (\text{recvmsg } rreq_rrep_sn)"

and "\{\text{PRreq:-2}\}[\lambda \xi. \xi'(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_{\text{AODV}} pp"

and "\text{Suc} 0 \leq \text{osn} \xi"

and "\forall dip \in \text{kD} (\text{rt} \xi). \text{Suc} 0 \leq \text{the} (\text{dhops} (\text{rt} \xi) ip)"

show "\forall dip \in \text{kD} (\text{rt} \xi). \text{rt} \xi \subseteq dip \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 "\text{ip} \in \text{kD} (\text{rt} \xi)"

moreover with * have "1 \leq \text{the} (\text{dhops} (\text{rt} \xi) ip)" by simp

moreover from \Suc 0 \leq \text{osn} \xi

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 \subseteq dip \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'\n
assume "((\xi, pp) \in \text{reachable } (\text{paodv } i ) \ (\text{recvmsg } rreq_rrep_sn)"
and \( \{ \text{PRrep}\} [ \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)) \]

\( p' \in \text{sterms} \Gamma_{AODV} \text{pp} \)

and \( "\text{Suc} 0 \leq \text{dsn} \xi" \)

and \( "\forall \text{ip} \in \text{kD} (\text{rt} \xi). \text{Suc} 0 \leq \text{the} (\text{dhops} (\text{rt} \xi, \text{ip}))" \)

show \( "\forall \text{ip} \in \text{kD} (\text{rt} \xi). \text{rt} \xi \sqsubseteq \text{ip} \text{update} (\text{rt} \xi) (\text{dip} \xi) (\text{dsn} \xi, \text{kno}, \text{val}, \text{Suc} (\text{hops} \xi), \text{sip} \xi)" \)

proof

fix \text{ip}

assume \( "\text{ip} \in \text{kD} (\text{rt} \xi)" \)

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 "\( \text{rt} \xi \sqsubseteq \text{ip} \text{update} (\text{rt} \xi) (\text{dip} \xi) (\text{dsn} \xi, \text{kno}, \text{val}, \text{Suc} (\text{hops} \xi), \text{sip} \xi)" \)

by (rule \text{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 \Rightarrow state \Rightarrow bool"

where "quality_increases \xi \xi' \equiv (\forall \text{dip} \in \text{kD} (\text{rt} \xi) \rightarrow \text{dip} \in \text{kD} (\text{rt} \xi')) \land \text{rt} \xi \sqsubseteq \text{dip} \text{rt} \xi' \land (\forall \text{dip}. \text{sqn} (\text{rt} \xi) \text{dip} \leq \text{sqn} (\text{rt} \xi') \text{dip})"

lemma quality_increasesI [intro!]:

assumes "(\forall \text{dip}. \text{dip} \in \text{kD} (\text{rt} \xi) \Rightarrow \text{dip} \in \text{kD} (\text{rt} \xi'))"

and "(\forall \text{dip}. \text{dip} \in \text{kD} (\text{rt} \xi'); \text{rt} \xi \sqsubseteq \text{dip} \text{rt} \xi'; \text{sqn} (\text{rt} \xi) \text{dip} \leq \text{sqn} (\text{rt} \xi') \text{dip}) \rightarrow \text{R dip} \xi \xi'"

shows "quality_increases \xi \xi'"

unfolding quality_increases_def using assms by clarsimp

lemma quality_increasesE [elim]:

fixes \text{dip}

assumes "quality_increases \xi \xi'"

and "\text{dip} \in \text{kD} (\text{rt} \xi)"

and "(\forall \text{dip}. \text{dip} \in \text{kD} (\text{rt} \xi'); \text{rt} \xi \sqsubseteq \text{dip} \text{rt} \xi'; \text{sqn} (\text{rt} \xi) \text{dip} \leq \text{sqn} (\text{rt} \xi') \text{dip}) \rightarrow \text{R dip} \xi \xi'"

shows "R dip \xi \xi'"

using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_rt_fresherD [dest]:

fixes \text{ip}

assumes "quality_increases \xi \xi'"

and "\text{ip} \in \text{kD} (\text{rt} \xi)"

shows "\text{rt} \xi \sqsubseteq \text{ip} \text{rt} \xi'"

using assms by auto

lemma quality_increases_sqnE [elim]:

fixes \text{dip}

assumes "quality_increases \xi \xi'"

and "\text{sqn} (\text{rt} \xi) \text{dip} \leq \text{sqn} (\text{rt} \xi') \text{dip} \Rightarrow \text{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'

assumes "\text{rt} (\sigma i) \sqsubseteq \text{dip} \text{rt} (\sigma \text{nhip})"

and qinc: "quality_increases (\sigma \text{nhip}) (\sigma' \text{nhip})"
and \( \text{dip} \in kD(rt (\sigma \text{nhip})) \)
shows \( rt (\sigma i) \sqsubseteq_{\text{dip}} rt (\sigma' \text{nhip}) \)

proof -
from \( \text{qinc} \) have \( rt (\sigma \text{nhip}) \sqsubseteq_{\text{dip}} rt (\sigma' \text{nhip}) \) using \( \text{dip} \in kD(rt (\sigma \text{nhip})) \)
by auto
with \( rt (\sigma i) \sqsubseteq_{\text{dip}} rt (\sigma \text{nhip}) \) show \( \text{thesis} \) ..
qed

lemma \( kD\_quality\_increases \) [elim]:
assumes \( "i \in kD(rt (\xi)" \)
and \( "\text{quality}\_increases \xi \xi'" \)
shows \( "i \in kD(rt (\xi')" \)
using \( \text{assms} \) by auto

lemma \( kD\_nsqn\_quality\_increases \) [elim]:
assumes \( "i \in kD(rt (\xi)" \)
and \( "\text{quality}\_increases \xi \xi'" \)
shows \( "i \in kD(rt (\xi')" \)
using \( \text{assms} \) by \( \text{rule} kD\_nsqn\_quality\_increases \) [THEN conjunct2]

lemma \( kD\_nsqn\_quality\_increases\_trans \) [elim]:
assumes \( "i \in kD(rt (\xi)" \)
and \( "s \leq \text{nsqn} (rt (\xi) i)" \)
and \( "\text{quality}\_increases \xi \xi'" \)
shows \( "i \in kD(rt (\xi')" \)
proof -
from \( \text{assms}(1-2) \) have \( "\text{nsqn} (rt (\xi) i) \leq \text{nsqn} (rt (\xi') i)" \) ..
next
from \( \text{assms}(1-2) \) have \( "\text{nsqn} (rt (\xi) i) \leq \text{nsqn} (rt (\xi') i)" \) ..
with \( s \leq \text{nsqn} (rt (\xi) i) \) show \( "s \leq \text{nsqn} (rt (\xi') i)" \) by \( \text{rule} \text{le}\_\text{trans} \)
qed

lemma \( \text{nsqn}\_quality\_increases\_nsqn\_lt\_lt \) [elim]:
assumes \( "i \in kD(rt (\xi)" \)
and \( "\text{quality}\_increases \xi \xi'" \)
and \( "s < \text{nsqn} (rt (\xi) i)" \)
shows \( "s < \text{nsqn} (rt (\xi') i)" \)
proof -
from \( \text{assms}(1-2) \) have \( "\text{nsqn} (rt (\xi) i) \leq \text{nsqn} (rt (\xi') i)" \) ..
with \( s < \text{nsqn} (rt (\xi) i) \) show \( "s < \text{nsqn} (rt (\xi') i)" \) by \( \text{simp} \)
qed

lemma \( \text{nsqn}\_quality\_increases\_dhops \) [elim]:
assumes \( "i \in kD(rt (\xi)" \)
and \( "\text{quality}\_increases \xi \xi'" \)
and \( "\text{nsqn} (rt (\xi) i) = \text{nsqn} (rt (\xi') i)" \)
shows \( "\text{the} (\text{dhops} (rt (\xi) i)) \geq \text{the} (\text{dhops} (rt (\xi') i))" \)
using \( \text{assms} \) unfolding \( \text{quality}\_increases\_def \)
by \( \text{clarsimp} \) \( \text{drule}(1) \) \text{bspec, clarsimp simp: rt\_fresher\_def2} \)

lemma \( \text{nsqn}\_quality\_increases\_nsqn\_eq\_le \) [elim]:
assumes \( "i \in kD(rt (\xi)" \)
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 "I ≤ 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 ": (sn < 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 \(\text{ip} \in kD(rt (σ' sip))\) show \(?\text{thesis}\)
proof (rule vD_or_iD)
  assume "ip \in iD(rt (σ' sip))"
  hence "the (flag (rt (σ' sip)) ip) = inv" ..
  with \(\text{sn} = \text{nsqn} (rt (σ' sip)) ip\) show \(?\text{thesis}\)
  by simp
next

  assume "ip \in vD(rt (σ' sip))"
  hence "nsqn (rt (σ' sip)) ip = sqn (rt (σ' sip)) ip" ..
  with \(\text{sn} = \text{nsqn} (rt (σ' sip)) ip\) have "nsqn (rt (σ' sip)) ip ≥ sqn (rt (σ' sip)) ip" by simp

  with \(\text{sn} = \text{nsqn} (rt (σ' sip)) ip\) ≤ sqn (rt (σ' sip)) ip" by simp

  with \(\text{sn} = \text{nsqn} (rt (σ' sip)) ip\) have "nsqn (rt (σ' sip)) ip > sn" by simp

  thus \(?\text{thesis}\) ..
qed

thus \(?\text{thesis}\) by (metis (mono_tags) le_cases not_le)
qed

with \(\text{ip} \in kD (rt (σ' sip))\) show "ip \in kD (rt (σ' sip)) ∧ \(?\text{nsqa}\)" ..
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 \in 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 \in 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 \(\text{assms}(1)\) have "quality_increases (σ sip) (σ' sip)" ..
  thus \(?\text{thesis}\) using \(\text{assms}(2-3)\) 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 \(\text{assms}\) 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 \in 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 \in kD(rt (σ sipc)) ∧ nsqn (rt (σ sipc)) dipc ≥ dsnc
  ∧ (nsqn (rt (σ sipc)) dipc = dsnc
  → (hopsc ≥ the (dhops (rt (σ sipc)) dipc)"
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 m =⇒ rreq_rrep_sn m"
by (cases m) simp_all

lemma recv_msg_fresh_inc_sn [simp, elim]:
"orecvmsg (msg_fresh) m =⇒ recvmsg 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 assms(1) and ⟨sip ≠ oip⟩ 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 ∈ iD(?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 "\(\text{sip} \neq \text{oip}\)"

with \(\text{assms}(1)\) have "\(\text{osn} \leq \text{sqn} \?\text{rt} \\text{oip}\)" by auto

thus "\(\text{osn} \leq \text{nsqn} \?\text{rt} \\text{oip}\)"

proof (rule \text{nat_le_eq_or_lt})

assume "\(\text{osn} < \text{sqn} \?\text{rt} \\text{oip}\)"

hence "\(\text{osn} \leq \text{sqn} \?\text{rt} \\text{oip} - 1\)" by simp

also have "\(\ldots \leq \text{nsqn} \?\text{rt} \\text{oip}\)" by (rule sqn_nsqn)

finally show "\(\text{osn} \leq \text{nsqn} \?\text{rt} \\text{oip}\)".

next

assume "\(\text{osn} = \text{sqn} \?\text{rt} \\text{oip}\)"

with \(\text{assms}(1)\) and \(\langle \text{sip} \neq \text{oip} \rangle\) have "\(\text{oip} \in \text{kD} \?\text{rt}\)"

and "\(\text{the (flag } \?\text{rt} \\text{oip}) = \text{val}\)"

by auto

hence "\(\text{nsqn} \?\text{rt} \\text{oip} = \text{sqn} \?\text{rt} \\text{oip}\) ..

with \(\langle \text{osn} = \text{sqn} \?\text{rt} \\text{oip} \rangle\) have "\(\text{nsqn} \?\text{rt} \\text{oip} = \text{oip}\)" by simp

thus "\(\text{osn} \leq \text{nsqn} \?\text{rt} \\text{oip}\)" by simp

qed

qed simp

qed

lemma \text{rrep_nsqn_is_fresh} [simp]:

fixes \(\sigma\) \(\text{msg} \) \(\text{hops} \) \(\text{dip} \) \(\text{dsn} \) \(\text{oip} \) \(\text{sip}\)

assumes "\(\text{rreq_rrep_fresh } \?\text{rt} \?\text{rt}\)"

shows "\(\text{msg_fresh } \?\text{msg}\)"

(proof Direct proof)

let \(?\text{rt} = \langle \text{rt} \?\text{rt}\rangle\)

from \(\text{assms}\) have "\(\text{sip} \neq \text{dip} \rightarrow \text{dip} \in \text{kD} \?\text{rt}\)\) \(\land \text{sqn} \?\text{rt} \?\text{dip} = \text{dsn} \land \text{the (flag } \?\text{rt} \?\text{dip}) = \text{val}\)"

by \text{simp}

hence "\(\text{sip} \neq \text{dip} \rightarrow \text{dip} \in \text{kD} \?\text{rt}\)\) \(\land \text{nsqn} \?\text{rt} \?\text{dip} \geq \text{dsn}\)"

by \text{clarsimp}

with \(\text{assms}\) show "\(\text{msg_fresh } \?\text{msg}\)"

by \text{clarsimp}

qed

lemma \text{rerr_nsqn_is_fresh} [simp]:

fixes \(\sigma\) \(\text{msg} \) \(\text{dests} \) \(\text{sip}\)

assumes "\(\text{rerr_invalid } \?\text{rt} \?\text{rt}\)"

shows "\(\text{msg_fresh } \?\text{msg}\)"

(proof Direct proof)

let \(?\text{rt} = \langle \text{rt} \?\text{rt}\rangle\)

from \(\text{assms}\) have "\(\forall \?\text{rip} \in \text{dom(dests)}. (\text{rip} \in \text{iD} \?\text{rt} \?\text{rt}) \land \text{sqn} \?\text{rt} \?\text{dip} = \text{dsn} \land \text{the (dests } \?\text{rip}) = \text{sqn} \?\text{rt} \?\text{rip})\)"

by \text{clarsimp}

have "\(\forall \?\text{rip} \in \text{dom(dests)}. (\?\text{rip} \in \text{kD} \?\text{rt} \?\text{rt}) \land \text{the (dests } \?\text{rip}) - 1 \leq \text{nsqn} \?\text{rt} \?\text{rip})\)"

by \text{clarsimp}

proof

fix \?\text{rip}

assume "\?\text{rip} \in \text{dom dests}"

with \?\text{rip} have "\?\text{rip} \in \text{iD} \?\text{rt} \?\text{rt}) \land \text{the (dests } \?\text{rip}) = \text{sqn} \?\text{rt} \?\text{rip})"

by \text{auto}

from \text{this(2)} have "\(\text{the (dests } \?\text{rip}) - 1 = \text{sqn} \?\text{rt} \?\text{rip} - 1\)" by simp

also have "\(\ldots \leq \text{nsqn} \?\text{rt} \?\text{rip}) \?\text{rip}\)" by (rule sqn_nsqn)

finally have "\(\text{the (dests } \?\text{rip}) - 1 \leq \text{nsqn} \?\text{rt} \?\text{rip})\)".

with \?\text{rip} \in \text{iD} \?\text{rt} \?\text{rt})

show "\?\text{rip} \in \text{kD} \?\text{rt} \?\text{rt}) \land \text{the (dests } \?\text{rip}) - 1 \leq \text{nsqn} \?\text{rt} \?\text{rip})"

by \text{clarsimp}
thus \( \text{"msg\_fresh } \sigma \? \text{msg"} \)
by simp
qd

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 dsnk osn sip
assume [simp]: \( m = \text{Rreq hops rreqid dip dsnk osn sip} \)
and \( \"msg\_fresh \sigma' m" \)
then have \( \text{"osn } \geq \text{1"} \) and \( \text{"sip } = \text{oip } \lor \text{(oip} \in \text{kD(rt (} \sigma \text{ sip)) } \land \text{nsqn (rt (} \sigma \text{ sip)) oip } \land \text{(nsqn (rt (} \sigma \text{ sip)) oip = osn } \rightarrow \text{(the (dhops (rt (} \sigma \text{ sip)) oip) } \leq \text{hops } \lor \text{the (flag (rt (} \sigma \text{ sip)) oip) = inv))}" \)
by auto
from this(2) show \( \text{"thesis} \)
proof
assume \( \text{"sip } = \text{oip"} \) with \( \text{"osn } \geq \text{1} \) show \( \text{"thesis"} \) by simp
next
assume \( \text{"oip} \in \text{kD(rt (} \sigma \text{ sip)) } \land \text{nsqn (rt (} \sigma \text{ sip)) oip } \land \text{(nsqn (rt (} \sigma \text{ sip)) oip = osn } \rightarrow \text{(the (dhops (rt (} \sigma \text{ sip)) oip) } \leq \text{hops } \lor \text{the (flag (rt (} \sigma \text{ sip)) oip) = inv))}" \)
moreover from qinc have \( \text{"quality\_increases } (\sigma) (\sigma' \text{ sip})" \)
ultimately have \( \text{"oip} \in \text{kD(rt (} \sigma \text{ sip)) } \land \text{nsqn (rt (} \sigma \text{ sip)) oip } \land \text{(nsqn (rt (} \sigma \text{ sip)) oip = osn } \rightarrow \text{(the (dhops (rt (} \sigma \text{ sip)) oip) } \leq \text{hops } \lor \text{the (flag (rt (} \sigma \text{ sip)) oip) = inv))}" \)
using \( \text{"osn } \geq \text{1} \) by (rule quality_increases_rreq_rrep_props [rotated 2])
with \( \text{"osn } \geq \text{1} \) show \( \"msg\_fresh \sigma' m" \)
by (clarsimp)
qd
next
fix hops dip dsnk oip sip
assume [simp]: \( m = \text{Rrep hopd dip dsnk oip sip} \)
and \( \"msg\_fresh \sigma' m" \)
then have \( \text{"dsn } \geq \text{1"} \) and \( \text{"sip } = \text{dip } \lor \text{(dip} \in \text{kD(rt (} \sigma \text{ sip)) } \land \text{dsn } \leq \text{nsqn (rt (} \sigma \text{ sip)) dip } \land \text{(nsqn (rt (} \sigma \text{ sip)) dip = dsnk } \rightarrow \text{(the (dhops (rt (} \sigma \text{ sip)) dip) } \leq \text{hops } \lor \text{the (flag (rt (} \sigma \text{ sip)) dip) = inv))}" \)
by auto
from this(2) show \( \text{"thesis} \)
proof
assume \( \text{"sip } = \text{dip"} \) with \( \text{"dsn } \geq \text{1} \) show \( \text{"thesis"} \) by simp
next
assume \( \text{"dip} \in \text{kD(rt (} \sigma \text{ sip)) } \land \text{dsn } \leq \text{nsqn (rt (} \sigma \text{ sip)) dip } \land \text{(nsqn (rt (} \sigma \text{ sip)) dip = dsnk } \rightarrow \text{(the (dhops (rt (} \sigma \text{ sip)) dip) } \leq \text{hops } \lor \text{the (flag (rt (} \sigma \text{ sip)) dip) = inv))}" \)
moreover from qinc have \( \text{"quality\_increases } (\sigma) (\sigma' \text{ sip})" \)
ultimately have \( \text{"dip} \in \text{kD(rt (} \sigma \text{ sip)) } \land \text{dsn } \leq \text{nsqn (rt (} \sigma \text{ sip)) dip } \land \text{(nsqn (rt (} \sigma \text{ sip)) dip = dsnk } \rightarrow \text{(the (dhops (rt (} \sigma \text{ sip)) dip) } \leq \text{hops } \lor \text{the (flag (rt (} \sigma \text{ sip)) dip) = inv))}" \)
using \( \text{"dsn } \geq \text{1} \) by (rule quality_increases_rreq_rrep_props [rotated 2])
with \( \text{"dsn } \geq \text{1} \) show \( \"msg\_fresh \sigma' m" \)
by clarsimp
qd
next
fix desnts 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

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 σ_AODV' :: "((ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp)) set"
where "σ_AODV' ≡ { (λ i. aodv_init i, Γ_AODV PAodv) }"

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

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

lemma oaodv_control_within [simp]: "control_within Γ_AODV (init (opaodv i))"
unfolding σ_AODV'_def by (rule control_withinI) (auto simp del: Γ_AODV_sims)

lemma σ_AODV'_labels [simp]: "((σ, p) ∈ σ_AODV' ) ⇒ labels Γ_AODV p = {PAodv←:0}"
unfolding σ_AODV'_def by simp

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

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

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

declare oseq_invariant_ctermsI [OF aodv_wf oaodv_control_within aodv_simple_labels aoav_trans, cterms_intros]
oseq_step_invariant_ctermsI [OF aodv_wf aoav_control_within aodv_simple_labels aoav_trans, cterms_intros]
end
theory C_Global_Invariants
imports C_Seq_Invariants
C_Aodv_Predicates
C_Fresher
C_Quality_Increases
"../../../AWN/OAWN_Convert"
C_OAodv
begin

lemma other_quality_increases [elim]:
  assumes "other quality_increases I σ σ'"
  shows "∀ j. quality_increases (σ j) (σ' j)"
  using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
  fixes m
  assumes *: "otherwith P I (orecvmsg Q) σ σ' a"
      and weakenP: "⋀ σ m. P σ m =⇒ P' σ m"
      and weakenQ: "⋀ σ m. Q σ m =⇒ Q' σ m"
  shows "otherwith P' I (orecvmsg Q') σ σ' a"
  proof
    fix j
    assume "j /∈ I"
    with * have "P (σ j) (σ' j)" by auto
    thus "P' (σ j) (σ' j)" by (rule weakenP)
  next
    from * have "orecvmsg Q σ a" by auto
    thus "orecvmsg Q' σ a" by (erule weakenQ)
  qed

lemma oreceived_msg_inv:
  assumes other: "⋀ σ σ' m. [ [ P σ m; other Q {i} σ σ' ] ] =⇒ P σ' m"
      and local: "⋀ σ m. P σ m =⇒ P (σ(i := σ i(msg := m)) m)"
  shows "opaodv i = (otherwith Q {i} (orecvmsg P), other Q {i} →) onl Γ_AODV (λ(σ, l). l ∈ {PAodv-:1} −→ P σ (msg (σ i)))"
  proof (inv_cterms, intro impI)
    fix σ σ' l
    assume "l = PAodv-:1 −→ P σ (msg (σ i))"
    and "l = PAodv-:1"
    and "other Q {i} σ σ'"
    from this(1-2) have "P σ (msg (σ i))" ..
    hence "P σ' (msg (σ i))" using 'other Q {i} σ σ'
        by (rule other)
    moreover from 'other Q {i} σ σ' have "σ' i = σ 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:
  "paodv i \models (recvmsg rreq_rrep_sn \to) onll \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). quality_increases \xi \xi')"
proof (rule step_invariantI)
  fix s a s'
  assume sr: "s \in reachable (paodv i) (recvmsg rreq_rrep_sn)" 
  and tr: "(s, a, s') \in trans (paodv i)" 
  and rm: "recvmsg rreq_rrep_sn a"
  from sr have srTT: "s \in reachable (paodv i) TT" ..
  from route_tables_fresher sr tr rm
  have "onll \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). quality_increases \xi \xi')" 
    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) \leq sqn (rt \xi')) (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_aodv aoqv_trans aodv_trans, 
    simplified seqll_onll_swap]

lemma quality_increases:
  "opaodv i \models (act (recvmsg rreq_rrep_sn), other A {i} \to) onll \Gamma_{AODV} (\lambda((\sigma, _), a, (\sigma', _)). quality_increases \sigma a (\sigma'))"
(is "\models (\?S, _ \to) _")
proof (rule onll_ostep_invariantI, simp)
  fix \sigma \tau l a \sigma' l' 
  assume or: "((\sigma, \tau), a) \in reachable (opaodv i) ?S (other A {i})" 
  and ll: "l \in labels \Gamma_{AODV} \tau" 
  and "\?S \tau a" 
  and tr: "((\sigma, \tau), a, (\sigma', \tau')) \in oseqp_sos \Gamma_{AODV} i" 
  and ll': "l' \in labels \Gamma_{AODV} \tau'" 
  from this(1-3) have "recvmsg (\lambda_. rreq_rrep_sn) \sigma a" 
    by (auto dest!: otherwith_actionD) 
  with or have orw: "((\sigma, \tau), a) \in reachable (opaodv i) (act (recvmsg rreq_rrep_sn)) 
    (other_quality_increases {i})" 
    otherwise weakenE [where QS="act (recvmsg rreq_rrep_sn)" 
    and QU="other_quality_increases {i}"]
  with tr ll ll' and recvmsg (\lambda_. rreq_rrep_sn) \sigma a have "quality_increases \sigma a (\sigma')" 
    by (drule onll_ostep_invariantD [OF olocal_quality_increases], auto simp: seqll_def) 
  with orw show "\forall j. quality_increases \sigma j (\sigma')" 
    by (auto dest!: otherwith_syncD)
qed

lemma rreq_rrep_nsqn_fresh_any_step_invariant:
  "opaodv i \models (act (recvmsg rreq_rrep_sn), other A {i} \to) onll \Gamma_{AODV} (\lambda((\sigma, _), a, _). anycast (msg_fresh \sigma) a)"
proof (rule ostep_invariantI, simp del: act_simp)
  fix \sigma \tau l a \sigma' l' 
  assume or: "((\sigma, \tau), a) \in reachable (opaodv i) (act (recvmsg rreq_rrep_sn)) (other A {i})" 
  and tr: "((\sigma, \tau), a, (\sigma', \tau')) \in oseqp_sos \Gamma_{AODV} i" 
  and recv: "act (recvmsg rreq_rrep_sn) \sigma a" 
  obtain l l' where "l \in labels \Gamma_{AODV} \tau" and "l' \in labels \Gamma_{AODV} \tau'" 
    by (metis aodv_ex_label)
  with or show "\forall j. quality_increases \sigma a (\sigma')" 
    by (auto dest!: otherwith_actionD) 
  with tr ll ll' and recvmsg (\lambda_. rreq_rrep_sn) \sigma a have "quality_increases \sigma a (\sigma')" 
    by (drule onll_ostep_invariantD [OF olocal_quality_increases], auto simp: seqll_def) 
  with or have orw: "((\sigma, \tau), a) \in reachable (opaodv i) (act (recvmsg rreq_rrep_sn)) 
    (other_quality_increases {i})" 
    otherwise weakenE [where QS="act (recvmsg rreq_rrep_sn)" 
    and QU="other_quality_increases {i}"]
  with orw show "\forall j. quality_increases \sigma a (\sigma')" 
    by (auto dest!: otherwith_syncD)
qed
from \(((\sigma, p), a, (\sigma', p')) \in \text{oseq}_{p} \text{soc} \Gamma_{AODV} i\)

have tr: "\(((\sigma, p), a, (\sigma', p')) \in \text{trans} (\text{opaodv} i)" by simp

have "anycast \((\text{rreq_rrep_fresh} \ (\text{rt} \ ((\sigma \ i))) \ a)\)"
proof -
  have "\text{opaodv} i \models_{A} (\text{act} \ (\text{recvmsg} \ \text{rreq_rrep_sn}), \text{other} \ A \ \{i\} \rightarrow)"
    \ \ \ \text{onll} \ \Gamma_{AODV} \ \text{(seqll} \ i \ (\lambda((\xi, \_), a, \_). \text{anycast} \ (\text{rreq_rrep_fresh} \ (\text{rt} \ \xi)) \ a))" by (rule \text{ostep_invariant\_weakenE} \ [OF \text{open\_seq\_step\_invariant} \ [OF \text{rreq\_rrep\_fresh\_any\_step\_invariant} \ \text{initiali\_aodv}, \text{simplified seqll\_onll\_swap}]) auto
  hence "\text{onll} \ \Gamma_{AODV} \ \text{(seqll} \ i \ (\lambda((\xi, \_), a, \_). \text{anycast} \ (\text{rreq_rrep_fresh} \ (\text{rt} \ \xi)) \ a)) \ (((\sigma, p), a, (\sigma', p')))"
using or tr recv by - (erule(4) \text{ostep\_invariantE})
thus \?thesis
using \langle \ l \in \text{labels} \ \Gamma_{AODV} \ p \rangle \ \text{and} \ \langle \ l' \in \text{labels} \ \Gamma_{AODV} \ p' \rangle \ \text{by} \ \text{auto}
qed

moreover have "anycast \((\text{rerr\_invalid} \ (\text{rt} \ ((\sigma \ i))) \ a)\)"
proof -
  have "\text{opaodv} i \models_{A} (\text{act} \ (\text{recvmsg} \ \text{rreq_rrep_sn}), \text{other} \ A \ \{i\} \rightarrow)"
    \ \ \ \text{onll} \ \Gamma_{AODV} \ \text{(seqll} \ i \ (\lambda((\xi, \_), a, \_). \text{anycast} \ (\text{rerr\_invalid} \ (\text{rt} \ \xi)) \ a))" by (rule \text{ostep_invariant\_weakenE} \ [OF \text{open\_seq\_step\_invariant} \ [OF \text{rerr\_invalid\_any\_step\_invariant} \ \text{initiali\_aodv}, \text{simplified seqll\_onll\_swap}]) auto
  hence "\text{onll} \ \Gamma_{AODV} \ \text{(seqll} \ i \ (\lambda((\xi, \_), a, \_). \text{anycast} \ (\text{rerr\_invalid} \ (\text{rt} \ \xi)) \ a)) \ (((\sigma, p), a, (\sigma', p')))"
using or tr recv by - (erule(4) \text{ostep\_invariantE})
thus \?thesis
using \langle \ l \in \text{labels} \ \Gamma_{AODV} \ p \rangle \ \text{and} \ \langle \ l' \in \text{labels} \ \Gamma_{AODV} \ p' \rangle \ \text{by} \ \text{auto}
qed

moreover have "anycast \((\text{rreq\_rrep\_sn} \ a)\)"
proof -
  from or tr recv
  have "\text{onll} \ \Gamma_{AODV} \ \text{(seqll} \ i \ (\lambda(\_, a, \_). \text{anycast} \ \text{rreq\_rrep\_sn} \ a)) \ (((\sigma, p), a, (\sigma', p')))"
by (rule \text{ostep\_invariantE} \ [OF \text{open\_seq\_step\_invariant} \ [OF \text{rreq\_rrep\_sn\_any\_step\_invariant} \ \text{initiali\_aodv}, \text{simplified seqll\_onll\_swap}]) auto
  thus \?thesis
using \langle \ l \in \text{labels} \ \Gamma_{AODV} \ p \rangle \ \text{and} \ \langle \ l' \in \text{labels} \ \Gamma_{AODV} \ p' \rangle \ \text{by} \ \text{auto}
qed

moreover have "anycast \((\lambda m. \text{not\_Pkt} m \rightarrow \text{msg\_sender} m = i) \ a)"
proof -
  have "\text{opaodv} i \models_{A} (\text{act} \ (\text{recvmsg} \ \text{rreq\_rrep\_sn}), \text{other} \ A \ \{i\} \rightarrow)"
    \ \ \ \text{onll} \ \Gamma_{AODV} \ \text{(seqll} \ i \ (\lambda(\xi, \_), a, \_). \text{anycast} \ (\lambda m. \text{not\_Pkt} m \rightarrow \text{msg\_sender} m = i) \ a))" by (rule \text{ostep\_invariant\_weakenE} \ [OF \text{open\_seq\_step\_invariant} \ [OF \text{sender\_ip\_valid} \ \text{initiali\_aodv}, \text{simplified seqll\_onll\_swap}]) auto
  thus \?thesis using or tr recv \langle \ l \in \text{labels} \ \Gamma_{AODV} \ p \rangle \ \text{and} \ \langle \ l' \in \text{labels} \ \Gamma_{AODV} \ p' \rangle \ \text{by} \ \text{auto}
qed

ultimately have "anycast \((\text{msg\_fresh} \ \sigma) \ a)"
by (simp\_all add: \text{anycast\_def}
split: seq\_action.split\_asm \text{msg\_split\_asm} \text{simp\_all}
thus "\text{onll} \ \Gamma_{AODV} \ \text{(\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:
"\text{opaodv} i \models_{A} (\text{otherwith} \ \text{quality\_increases} \ \{i\} \ \text{orecvmsg} \ \text{msg\_fresh})", 308
lemma oquality_increases_nsqn_fresh:
  oosn_rreq:
  lemma rreq_sip:
  lemma:

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

lemma oquality_increases_nsqn_fresh:
  "opaodv i ⊦ (other with quality_increases {i} (orecmsg msg_fresh),
  other quality_increases {i} (i →)
  onl Γ AODV (λ(σ, _). 1 ∈ {PAodv-:4, PAodv-:5} → PReq-:n | n. True → 1 ≤ onl ξ))"
by (rule ostep_invariant_weakenE [OF oquality_increases]) auto

lemma oosn_rreq:
  "opaodv i ⊦ (other with quality_increases {i} (orecmsg msg_fresh),
  other quality_increases {i} (i →)
  onl Γ AODV (seq1 _ (λ(ξ, 1). 1 ∈ {PAodv-:4, PAodv-:5} ∪ {PReq-:n | n. True → 1 ≤ onl ξ}))"
by (rule oinvariant_weakenE [OF open_seq_invariant [OF oosn_rreq initiali_aodv]])
(auto simp: seq1_onl_swap)

lemma rreq_sip:
  "opaodv i ⊦ (other with quality_increases {i} (orecmsg msg_fresh),
  other quality_increases {i} (i →)
  onl Γ AODV (λ(σ, 1). (1 ∈ {PAodv-:4, PAodv-:5, PReq-:0, PReq-: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 aodv_trans
onl_oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
onl_oinvariant_sterms [OF aodv_wf oosn_rreq]
simp add: seq1simp
simp del: One_nat_def, rule impl)

fix σ σ' p l
assume "(σ, p) ∈ ooreachable (opaodv i) "?S "?U"
and "1 ∈ labels Γ AODV p"
and pre:
"(1 = PAodv-:4 ∨ 1 = PAodv-:5 ∨ 1 = PReq-:0 ∨ 1 = PReq-: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: "(1=PAodv-:4 ∨ 1=PAodv-:5 ∨ 1=PReq-:0 ∨ 1=PReq-: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 \( \text{oip} (\sigma \ i) \in kD (rt (\sigma' (sip (\sigma' i)))) \)
\( \wedge \ \text{osn} (\sigma' i) \leq \text{nsqn} (rt (\sigma' (sip (\sigma' i)))) (\text{oip} (\sigma' i)) \)
\( \wedge (\text{nsqn} (rt (\sigma' (sip (\sigma' i)))) (\text{oip} (\sigma' i)) = \text{osn} (\sigma' i) \)
\[ \rightarrow \text{the} (\text{dhops} (rt (\sigma' (sip (\sigma' i)))) (\text{oip} (\sigma' i))) \leq \text{hops} (\sigma' i) \]
\( \vee \text{the} (\text{flag} (rt (\sigma' (sip (\sigma' i)))) (\text{oip} (\sigma' i))) = \text{inv}'' \)

proof (cases "sip (\sigma i) = i")
asssume "\( \text{sip} (\sigma i) \neq i \)"
from (other quality increases \( \{i\} \sigma \sigma' \))
have "\( \text{quality_increases} (\sigma (\sigma i)) (\sigma' (\sigma i))) \)"
by (rule otherE) (clarsimp simp: \( \text{seql_onl_swap} \))

moreover from \( \langle (\sigma, p) \rangle \in \text{oreachable} \) (opaodv i) \( ?S ?U \) \( l \in \text{labels} \ \Gamma_{AODV} p \) and hyp
have "\( 1 \leq \text{osn} (\sigma i) \)"
by (auto dest!: \( \sigma i \neq i \) hyp' and pre)

ultimately show \( ?\text{thesis} \)
by (rule quality_increases_rreq_rrep Props)

next
assume "\( \text{sip} (\sigma i) = i \)" thus \( ?\text{thesis} \)
using \( \sigma i = \sigma i \) hyp and pre by auto

qed

(qed (auto elim!: quality_increases_rreq_rrep Props'))

leamda odn_rrep:
"opaodv i \models (\text{otherwith} \ \text{quality_increases} \ \{i\} \ \text{orecmsg} \ \text{msg_fresh}, \)
other quality increases \( \{i\} \rightarrow \)
\( \text{onl} \ \Gamma_{AODV} \ \text{seql} i (\lambda(l, 1). l \in \{\text{PAodv-6}, \text{PAodv-7}\} \cup \{\text{PRrep-\{n\} True} \rightarrow 1 \leq \text{dsn} \ \xi\}) \)"
by (rule \( \text{onl_invariant_weakenE} \) \( \text{DF open_seq_invariant} \) \( \text{DF dsn_rrep initiali_aodv} \))

(auto simp: \( \text{seql_onl_swap} \))

leamda rrep sip:
"opaodv i \models (\text{otherwith} \ \text{quality_increases} \ \{i\} \ \text{orecmsg} \ \text{msg_fresh}, \)
other quality increases \( \{i\} \rightarrow \)
\( \text{onl} \ \Gamma_{AODV} \ \lambda(\sigma, i). \)
\( (1 \in \{\text{PAodv-6}, \text{PAodv-7}, \text{PRrep-0}, \text{PRrep-1} \} \wedge \text{sip} (\sigma i) \neq \text{dip} (\sigma i)) \)
\[ \rightarrow \text{dip} (\sigma i) \in kD (rt (\sigma (\text{sip} (\sigma i)))) \]
\[ \wedge (\text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i)) \geq \text{dsn} (\sigma i)) \]
\[ \wedge (\text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i)) = \text{dsn} (\sigma i)) \]
\[ \rightarrow (\text{hops} (\sigma i) \geq \text{the} (\text{dhops} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i))) \]
\[ \vee \text{the} (\text{flag} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i))) = \text{inv}))'' \]

(is "\( _= (\{\text{S}, \text{U} \rightarrow _) \))"

proof (inv cterms inv add: \( \text{oseq_step_invariant_sterms} \) \( \text{DF quality_increases_nsqn_fresh aodv_wf} \)
aodv_trans]
\( \text{onl_invariant_sterms} \) \( \text{DF aodv_wf oreceived_rreq_rrep nsqn_fresh_inv} \)
onl_invariant_sterms \( \text{DF aodv_wf dsn_rrep} \)

(simp del: One_nat_def, rule impl)

fix \( \sigma \sigma' p l \)
assume "\( (\sigma, p) \in \text{oreachable} (opaodv i) \) \( ?S ?U \)"
and "\( l \in \text{labels} \ \Gamma_{AODV} p \)"
and pre:
"\( (1 = \text{PAodv-6} \lor 1 = \text{PAodv-7} \lor 1 = \text{PRrep-0} \lor 1 = \text{PRrep-1}) \wedge \text{sip} (\sigma i) \neq \text{dip} (\sigma i) \)
\[ \rightarrow \text{dip} (\sigma i) \in kD (rt (\sigma (\text{sip} (\sigma i)))) \]
\[ \wedge (\text{dsn} (\sigma i) \leq \text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i))) \]
\[ \wedge (\text{nsqn} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i)) = \text{dsn} (\sigma i)) \]
\[ \rightarrow (\text{the} (\text{dhops} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i))) \leq \text{hops} (\sigma i) \]
\[ \vee \text{the} (\text{flag} (rt (\sigma (\text{sip} (\sigma i)))) (\text{dip} (\sigma i))) = \text{inv})'' \]

and "\( \text{other quality increases} \ \{i\} \sigma' \sigma' \)"
and hyp: "\( (1=\text{PAodv-6} \lor 1=\text{PAodv-7} \lor 1=\text{PRrep-0} \lor 1=\text{PRrep-1}) \wedge \text{sip} (\sigma' i) \neq \text{dip} (\sigma' i)"
(is "\$\text{labels} \land \text{sip}(\sigma' i) \neq \text{dip}(\sigma i)\")

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

with hyp have hyp': "\$\text{labels} \land \text{sip}(\sigma i) \neq \text{dip}(\sigma i)\" by simp

show "\$\text{dip}(\sigma' i) \in kD(\text{rt}(\sigma' (\text{sip}(\sigma' i))))\land\text{dsn}(\sigma' i) \leq \text{nsq}(\text{rt}(\sigma' (\text{sip}(\sigma' i)))) (\text{dip}(\sigma' i))\land(\text{nsq}(\text{rt}(\sigma' (\text{sip}(\sigma' i)))) (\text{dip}(\sigma' i)) = \text{dsn}(\sigma' i)\rightarrow\text{the}(\text{d-hop}(\text{rt}(\sigma' (\text{sip}(\sigma' i)))) (\text{dip}(\sigma' i)) \leq \text{hops}(\sigma' i)\lor\text{the}(\text{flag}(\text{rt}(\sigma' (\text{sip}(\sigma' i)))) (\text{dip}(\sigma' i)) = \text{inv})"

proof (cases "\$\text{sip}(\sigma i) = \sigma i\")

assume "\$\text{sip}(\sigma i) \neq \sigma 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: \(\text{sip}(\sigma i) \neq \sigma i\))

moreover from \(\langle\sigma, p\rangle\in\text{oreachable} (\text{opaodv} i)\ ?S \ ?U \ i\in \text{labels} \ \Gamma_{\text{AODV} p}\) and hyp

have "\(1 \leq \text{dsn}(\sigma' i)\)"

by (auto dest!: onloinvariant_weakenD [OF odsn_rrep]

simp add: seqlsimp \(\sigma' i = \sigma i\))

moreover from \(\text{sip}(\sigma i) \neq \sigma i\) hyp' and pre

have "\$\text{dip}(\sigma' i) \in kD(\text{rt}(\sigma(\text{sip}(\sigma i))))\land\text{dsn}(\sigma' i) \leq \text{nsq}(\text{rt}(\sigma(\text{sip}(\sigma i)))) (\text{dip}(\sigma' i))\land(\text{nsq}(\text{rt}(\sigma(\text{sip}(\sigma i)))) (\text{dip}(\sigma' i)) = \text{dsn}(\sigma' i)\rightarrow\text{the}(\text{d-hop}(\text{rt}(\sigma(\text{sip}(\sigma i)))) (\text{dip}(\sigma' i)) \leq \text{hops}(\sigma' i)\lor\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) = \sigma i\" thus ?thesis

using \(\sigma' i = \sigma i\) hyp and pre by auto

qed

(auto simp add: seqlsimp elim!: quality increases rreq rrep props')

lemma rerr_sip:

"\(\text{opaodv i} \vdash (\text{otherwith quality increases \{i\} (orecvmsg msg_fresh), other quality increases \{i\} \rightarrow)\)\)

onl \(\Gamma_{\text{AODV}} \ (\lambda(\sigma, l) . l \in \{\text{PAodv-:8, PAodv-:9, PRerr-:0, PRerr-:1}\) \rightarrow (\forall \text{rip} \in \text{dom(dests}(\sigma i)). \text{rip} \in kD(\text{rt}(\sigma \text{sip}(\sigma i)))\land \text{the}(\text{dests}(\sigma i) \text{rip} - 1 \leq \text{nsq}(\text{rt}(\sigma \text{sip}(\sigma i))) \text{rip}))\)"

(is "\_ \vdash (?S, ?U \rightarrow \_ )")

proof -

\{ fix \text{dests rip sip rsn and} \(\sigma \sigma' :: \"\text{ip} \Rightarrow \text{state}\"

assume ginc: "\(\forall j. \text{quality increases}(\sigma j) (\sigma' j)\)"

and *: "\(\forall \text{rip} \in \text{dom dests}. \text{rip} \in kD(\text{rt}(\sigma \text{sip}))\land \text{the}(\text{dests}(\sigma i) \text{rip} - 1 \leq \text{nsq}(\text{rt}(\sigma \text{sip})) \text{rip})\)"

and "\text{dests rip} = \text{Some rsn}\"

from this(3) have "\text{rip} \in \text{dom dests}" by auto

with * and \(\text{dests rip} = \text{Some rsn}\) have \(\text{rip} \in kD(\text{rt}(\sigma \text{sip}))\) and \(\text{rsn} - 1 \leq \text{nsq}(\text{rt}(\sigma \text{sip})) \text{rip}\)

by (auto dest!: bspec)

from ginc have "quality increases(\sigma \text{sip}) (\sigma' \text{sip})" ..

have \(\text{rip} \in kD(\text{rt}(\sigma' \text{sip})) \land \text{rsn} - 1 \leq \text{nsq}(\text{rt}(\sigma' \text{sip})) \text{rip}\)

proof

from \(\text{rip} \in kD(\text{rt}(\sigma \text{sip}))\) and \(\text{quality increases}(\sigma \text{sip}) (\sigma' \text{sip})\)

show \(\text{rip} \in kD(\text{rt}(\sigma' \text{sip}))\) ..

next

from \(\text{rip} \in kD(\text{rt}(\sigma \text{sip}))\) and \(\text{quality increases}(\sigma \text{sip}) (\sigma' \text{sip})\)

have "\(\text{nsq}(\text{rt}(\sigma \text{sip})) \text{rip} \leq \text{nsq}(\text{rt}(\sigma' \text{sip})) \text{rip}\)" ..

with \(\text{rsn} - 1 \leq \text{nsq}(\text{rt}(\sigma \text{sip}) \text{rip})\) show \(\text{rsn} - 1 \leq \text{nsq}(\text{rt}(\sigma' \text{sip})) \text{rip}\)

by (rule le_trans)

qed

\} note partial = this

show ?thesis
by (inv_cterms inv add: oseq_step_invariant_sterms [OF equality_increases_nsqn_fresh aodv_wf aodv_trans]
onl_oinvariant_sterms [OF aodv_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: "paodv i \|= 
onl Γ_AODV (λ(ξ, l). (1 = 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 odests_vD_inc_sqn =
open_seq_invariant [OF dests_vD_inc_sqn initiali_aodv aodv_trans, simplified seql_onl_swap,
  THEN oinvariant_anyact]

lemmas oprerr_guard =
open_seq_invariant [OF prerr_guard initiali_aodv aodv_trans, simplified seql_onl_swap,
  THEN oinvariant_anyact]

Proposition 7.28

lemma seq_compare_next_hop':
"opaodv i \|= (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i} \→) onl Γ_AODV (λ(σ, _).
    ∀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)"
(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"
    (is "_ \|= (?S, ?U \→) _")
  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_other) 
  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)))" 
  "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)))) (0, unk, val, Suc 0, sip (σ i)))) dip)))" 
}
proof

fix dip

assume "dip∈kD(rt (σ i))"

with a1 and a2

have "\((\text{nhop} \ (rt \ (σ i)) \ \text{dip}) \neq \ \text{dip} \ \rightarrow \ \text{dip}\in\text{kD}(rt \ (σ' \ (\text{the} \ (\text{nhop} \ (rt \ (σ i)) \ \text{dip})))) \wedge nsqn \ (rt \ (σ i)) \ \text{dip} \leq nsqn \ (rt \ (σ' \ (\text{the} \ (\text{nhop} \ (rt \ (σ i)) \ \text{dip})))) \ \text{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) \neq dip → dip\in\text{kD}(rt \ (σ' \ (\text{the} \ (\text{nhop} \ (rt \ (σ i)) \ \text{dip})))) \wedge \text{nsqn} \ (rt \ (σ i)) \ \text{dip} ≤ \text{nsqn} \ (rt \ (σ' \ (\text{the} \ (\text{nhop} \ (rt \ (σ i)) \ \text{dip})))) \ \text{dip}"

and qinc: "∀ j. quality_increases (σ j) (σ' j)"

and*: "sip \neq oip → oip\in\text{kD}(rt \ (σ sip)) \wedge osn ≤ nsqn \ (rt \ (σ sip)) \ \text{oip} \wedge \text{nsqn} \ (rt \ (σ sip)) \ oip = osn → \ (\text{the} \ (\text{dhops} \ (rt \ (σ sip))) \ \text{oip}) ≤ \text{hops} \ \vee \ \ (\text{the} \ (\text{flag} \ (rt \ (σ sip))) \ \text{oip}) = \text{inv}"

from pre and qinc

have pre": "∀ dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) \neq dip → dip\in\text{kD}(rt \ (σ' \ (\text{the} \ (\text{nhop} \ (rt \ (σ i)) \ \text{dip})))) \wedge \text{nsqn} \ (rt \ (σ i)) \ \text{dip} ≤ \text{nsqn} \ (rt \ (σ' \ (\text{the} \ (\text{nhop} \ (rt \ (σ i)) \ \text{dip})))) \ \text{dip}

by (rule basic)

have "(the (nhop (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) oip) \neq oip → oip\in\text{kD}(rt \ (σ' \ (\text{the} \ (\text{nhop} \ (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip))) oip))) \wedge \text{nsqn} \ (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip)) oip ≤ \text{nsqn} \ (rt \ (σ' \ (\text{the} \ (\text{nhop} \ (update (rt (σ i)) oip (osn, kno, val, Suc hops, sip))) oip))) oip)"

(is "?\text{nhop_not_oip} → ?\text{oip_in_kD} \wedge ?\text{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) \neq oip"

with pre' show "?\text{oip_in_kD} \wedge ?\text{nsqn_le_nsqn}" by auto

next

assume rtnot: "rt (σ i) \neq 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 "?\text{oip_in_kD} \wedge ?\text{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) \neq dip → dip\in\text{kD}(rt \ (σ' \ (\text{the} \ (\text{nhop} \ (rt (σ i)) \ \text{dip})))) \wedge \text{nsqn} \ (rt (σ i)) \ \text{dip} ≤ \text{nsqn} \ (rt \ (σ' \ (\text{the} \ (\text{nhop} \ (rt (σ i)) \ \text{dip})))) \ \text{dip}"

and qinc: "∀ j. quality_increases (σ j) (σ' j)"

and*: "sip \neq oip → oip\in\text{kD}(rt \ (σ sip)) \wedge osn ≤ nsqn \ (rt \ (σ sip)) \ \text{oip} \wedge \text{nsqn} \ (rt \ (σ sip)) \ oip = osn → \ (\text{the} \ (\text{dhops} \ (rt \ (σ sip))) \ \text{oip}) ≤ \text{hops} \ \vee \ \ (\text{the} \ (\text{flag} \ (rt \ (σ sip))) \ \text{oip}) = \text{inv}"

from pre and qinc

have pre": "∀ dip∈kD (rt (σ i)). the (nhop (rt (σ i)) dip) \neq dip → dip\in\text{kD}(rt \ (σ' \ (\text{the} \ (\text{nhop} \ (rt (σ i)) \ \text{dip})))) \wedge \text{nsqn} \ (rt (σ i)) \ \text{dip} ≤ \text{nsqn} \ (rt \ (σ' \ (\text{the} \ (\text{nhop} \ (rt (σ i)) \ \text{dip})))) \ \text{dip}"

by (rule basic)
have \( \forall \text{dip} \in \mathcal{K}\mathcal{D}(\text{rt} (\sigma \ i)). \)
\( \langle \text{nhop} (\text{update} (\text{rt} (\sigma \ i)) \text{ oip} (\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip})) \text{ dip} \rangle \neq \text{dip} \)
\( \rightarrow \langle \text{dip} \in \mathcal{K}\mathcal{D}(\text{rt} (\sigma' (\text{the} (\text{nhop} (\text{update} (\text{rt} (\sigma \ i)) \text{ oip} (\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip})))) \text{ dip}))) \rangle \)
\( \wedge \text{nsqn} (\text{update} (\text{rt} (\sigma \ i)) \text{ oip} (\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip})) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma' (\text{the} (\text{nhop} (\text{update} (\text{rt} (\sigma \ i)) \text{ oip} (\text{osn}, \text{kno}, \text{val}, \text{Suc hops}, \text{sip})))) \text{ dip})))) \)
\( (\text{is} "\forall \text{dip} \in \mathcal{K}\mathcal{D}(\text{rt} (\sigma \ i)). \_ \rightarrow \text{dip} \in \mathcal{K}\mathcal{D}(\text{rt} (\sigma \ i)) \text{ dip} \wedge \text{nsqn} \text{ dip}" )\)

proof (intro ballI impI, split update rt_split_asm)
fix dip
assume "dip \in \mathcal{K}\mathcal{D}(\text{rt} (\sigma \ i))"
and "the (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 \mathcal{K}\mathcal{D}(\text{rt} (\sigma \ i)) \text{ dip} \wedge \text{nsqn} \text{ dip}" by simp

next
fix dip
assume "dip \in \mathcal{K}\mathcal{D}(\text{rt} (\sigma \ i))"
and notdip: "the (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 \mathcal{K}\mathcal{D}(\text{rt} (\sigma \ i)) \text{ dip} \wedge \text{nsqn} \text{ dip}" by simp
proof (cases "dip = \text{oip}")
assume "dip \neq \text{oip}"
with pre' ⟨dip \in \mathcal{K}\mathcal{D}(\text{rt} (\sigma \ i))⟩ notdip *
show ?thesis by clarsimp
next
assume "dip = \text{oip}"
with rtnot qinc ⟨dip \in \mathcal{K}\mathcal{D}(\text{rt} (\sigma \ i))⟩ notdip *
have "\text{nsqn} \text{ dip}" by simp (metis kD nsqn_quality_increases_trans)
moreover from ⟨dip = \text{oip}⟩ rtnot qinc ⟨dip \in \mathcal{K}\mathcal{D}(\text{rt} (\sigma \ i))⟩ notdip *
have "\text{nsqn} \text{ dip}" by simp (metis kD nsqn_quality_increases_trans)
ultimately show ?thesis..
qed

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_aodv,
simplified seql_onl_swap]

lemmas ozero_seq_unk_hops_one =
lemma orreachable_fresh_okD_unk_or_atleast_one:
  fixes dip
  assumes "(σ, p) ∈ orreachable (opaodv 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 ∈ labels Γ AODV p" by (metis aodv_ex_label)
  with assms(1)
  have "∀ dip ∈ kD (rt (σ i)). ?P dip" by - (drule oinvariant_weakenD [OF okD_unk_or_atleast_one [OF oaodv_trans aodv_trans]],
    auto dest!: otherwith_actionD onlD simp: seqlsimp)
  with ⟨dip ∈ kD (rt (σ i))⟩
  show ?thesis
  by simp
 qed

lemma orreachable_fresh_ozero_seq_unk_hops_one:
  fixes dip
  assumes "(σ, p) ∈ orreachable (opaodv 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 ∈ labels Γ AODV 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 Γ AODV (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 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 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 ov: "?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: "∃i. 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 - (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(l) 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_ifrefl)
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 (dhop (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

318
moreover from a3 have \( \text{dip} \in v_D(\text{rt}\ (\sigma\ (\text{the}\ (\text{nhop}\ (\text{rt}\ (\sigma\ i))\ \text{dip})))\))
using nochange and \( \forall j.\ \sigma\ j = \sigma' j \) by clarsimp
moreover from a4 have \( \text{the}\ (\text{nhop}\ (\text{rt}\ (\sigma\ i))\ \text{dip}) \neq \text{dip} \) by simp
ultimately have \( \text{rt}\ (\sigma\ i) \sqsubseteq_{\text{dip}} \text{rt}\ (\sigma\ (\text{the}\ (\text{nhop}\ (\sigma\ i))\ \text{dip})))\)
using pre by simp

then \( \text{rt}\ (\sigma\ i) \sqsubseteq_{\text{dip}} \text{rt}\ (\sigma\ (\text{the}\ (\text{nhop}\ (\sigma\ i))\ \text{dip})))\)
using \( \forall j.\ \sigma\ j = \sigma' j \) by simp
thus \"?thesis\" by simp

next
assume \( \text{change} : \text{rt1} \neq \text{rt}\ (\sigma\ i) \)
from after a2 have \( \text{dip} \in k_D(\text{rt}\ (\sigma\ i))\)
by auto
show ?thesis
proof (cases "\text{dip} = \text{oip}\")
  assume "\text{dip} \neq \text{oip}\"

  with a2 have \( \text{dip} \in v_D(\text{rt}\ (\sigma\ i))\)
  by auto
moreover with a3 a5 after and \( \text{dip} \neq \text{oip}\)
  have \( \text{dip} \in v_D(\text{rt}(\sigma\ (\text{the}(\text{nhop}(\text{rt}(\sigma\ i))\ \text{dip}))))\)
  by simp metis
moreover from a4 and \( \text{dip} \neq \text{oip}\) have \( \text{the}(\text{nhop}(\text{rt}(\sigma\ i))\ \text{dip}) \neq \text{dip} \) by simp
ultimately have \( \text{rt}\ (\sigma\ i) \sqsubseteq_{\text{dip}} \text{rt}\ (\sigma\ (\text{the}(\text{nhop}(\sigma\ i))\ \text{dip})))\)
using pre by simp

  with after and a5 and \( \text{dip} \neq \text{oip}\) show ?thesis
by simp (metis rt_strictly_fresher_update_other rt_strictly_fresher_irefl)

next
assume "\text{dip} = \text{oip}\"

with a4 and \( \text{change} \) have "\text{sip} \neq \text{oip}\" by simp
with a6 have "\text{oip} \in k_D(\text{rt}(\sigma\ \text{sip}))\"
  and "\text{osn} \leq \text{nsqn}\ (\text{rt}(\sigma\ \text{sip}))\ \text{oip}" by auto

from a3 change \( \text{dip} = \text{oip}\) have "\text{oip} \in v_D(\text{rt}(\sigma'\ \text{sip}))\" by simp
hence "\text{the}\ (\text{flag}\ (\text{rt}(\sigma'\ \text{sip}))\ \text{oip}) = \text{val}\" by simp

from \( \text{oip} \in k_D(\text{rt}(\sigma\ \text{sip}))\)
have "\text{osn} < \text{nsqn}\ (\text{rt}(\sigma'\ \text{sip}))\ \text{oip} \lor (\text{osn} = \text{nsqn}\ (\text{rt}(\sigma'\ \text{sip}))\ \text{oip}) \land (\text{the}\ (\text{dhops}\ (\text{rt}(\sigma'\ \text{sip}))\ \text{oip}) \leq \text{hops})\"
proof
  assume "\text{oip} \in v_D(\text{rt}(\sigma\ \text{sip}))\"
  hence "\text{the}\ (\text{flag}\ (\text{rt}(\sigma\ \text{sip}))\ \text{oip}) = \text{val}\" by simp
  with a6 \( \text{sip} \neq \text{oip}\) have "\text{nsqn}\ (\text{rt}(\sigma\ \text{sip}))\ \text{oip} = \text{osn} \rightarrow\kr(\text{the}(\text{dhops}(\text{rt}(\sigma\ \text{sip}))\ \text{oip}) \leq \text{hops})\"
by simp
show ?thesis
proof (cases "\text{sip} = \text{i}\")
  assume "\text{sip} \neq \text{i}\"
  with a5 have "\text{\sigma\ sip} = \text{\sigma'} \text{sip}\" by simp
  with \( \text{osn} \leq \text{nsqn}(\text{rt}(\sigma\ \text{sip}))\ \text{oip}\)
  and \( \text{\nsqn}(\text{rt}(\sigma\ \text{sip}))\ \text{oip} = \text{osn} \rightarrow\kr(\text{the}(\text{dhops}(\text{rt}(\sigma\ \text{sip}))\ \text{oip}) \leq \text{hops})\)
show ?thesis by auto

next
— alternative to using sip_not_ip
assume [simp]: "\text{sip} = \text{i}\"
have "\text{rt1} = \text{rt}\ (\sigma\ i)\"
proof (rule update_cases_kD, simp, all)
  from \( \text{Suc} 0 \leq \text{osn}\) show "0 < \text{osn}\" by simp
next
  from \( \text{oip} \in k_D(\text{rt}(\sigma\ \text{sip}))\) and \( \text{\sip} = \text{\i}\) show "\text{oip} \in k_D(\text{rt}(\sigma\ \text{i}))\"
  by simp
next
assume "\text{sqn}(\text{rt}(\sigma\ \text{i}))\ \text{oip} < \text{osn}\"
also from \( osn \leq nsqn \ (rt \ (\sigma \ sip)) \ oip \) have ". . . \leq nsqn \ (rt \ (\sigma \ i)) \ oip " by simp
also have ". . . \leq sqn \ (rt \ (\sigma \ i)) \ oip "
by (rule nsqn_sqn)
finally have "sqn \ (rt \ (\sigma \ i)) \ oip < sqn \ (rt \ (\sigma \ i)) \ oip " .
hence False by simp
thus "(\lambda a. if a = oip then Some \ (osn, kno, val, Suc \ hops, i) else rt \ (\sigma \ i) a) = rt \ (\sigma \ i) " ..

next
assume "sqn \ (rt \ (\sigma \ i)) \ oip = osn"
and "Suc \ hops < the \ (dhops \ (rt \ (\sigma \ i)) \ oip)"
from this(1) and \( oip \in vD \ (rt \ (\sigma \ sip)) \) have "nsqn \ (rt \ (\sigma \ i)) \ oip = osn"
by simp
with \( nsqn \ (rt \ (\sigma \ sip)) \ oip = osn \rightarrow \ the \ (dhops \ (rt \ (\sigma \ sip)) \ oip) \leq hops \) have "the \ (dhops \ (rt \ (\sigma \ i)) \ oip) \leq hops " by simp
with \( Suc \ hops < the \ (dhops \ (rt \ (\sigma \ i)) \ oip) \) have False by simp
thus "(\lambda a. if a = oip then Some \ (osn, kno, val, Suc \ hops, i) else rt \ (\sigma \ i) a) = rt \ (\sigma \ i) " ..

next
assume "the \ (flag \ (rt \ (\sigma \ i)) \ oip) = inv"
with \( the \ (flag \ (rt \ (\sigma \ sip)) \ oip) = val \) have False by simp
thus "(\lambda a. if a = oip then Some \ (osn, kno, val, Suc \ hops, i) else rt \ (\sigma \ i) a) = rt \ (\sigma \ i) " ..

next
from \( oip \in kD \ (rt \ (\sigma \ sip)) \)
show "(\lambda a. if a = oip then Some \ (the \ (rt \ (\sigma \ i) \ oip)) else rt \ (\sigma \ i) a) = rt \ (\sigma \ i) "
by (auto dest!: kD_Some)
qed
with change have False ..
thus ?thesis ..
qed

next
assume "oip \in iD \ (rt \ (\sigma \ sip))"
with \( the \ (flag \ (rt \ (\sigma \ sip)) \ oip) = val \) and a5 have "sip = i"
by (metis f.distinct(1) iD_flag_is_inv)
from \( oip \in iD \ (rt \ (\sigma \ sip)) \) have "the \ (flag \ (rt \ (\sigma \ sip)) \ oip) = inv " by auto
with \( sip = i; Suc \ 0 \leq osn; change \ after \ oip \in kD \ (rt \ (\sigma \ sip)) \)
have "nsqn \ (rt \ (\sigma \ sip)) \ oip < nsqn \ (rt \ (\sigma \ sip)) \ oip "
unfolding update_def
by (clarsimp split: option.split_asm split_if_asm)
(auto simp: sqn_def)
with \( osn \leq nsqn \ (rt \ (\sigma \ sip)) \ oip \) have "osn < nsqn \ (rt \ (\sigma \ sip)) \ oip "
by simp
thus ?thesis ..
qed
thus ?thesis
proof
assume osnlt: "osn < nsqn \ (rt \ (\sigma \ sip)) \ oip"
from \( dip \in kD \ (rt \ (\sigma \ i)) \) and \( dip = oip \) have "dip \in kD \ (?rt1) " by simp
moreover from a3 have "dip \in 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 \ (\sigma \ 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)

qed
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)" 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)" by (simp add: change)
    finally show ?thesis . qed
    moreover have "π5(the (?rt2 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 i) oip)) < π5(the (rt (σ i) oip))" by simp
      qed
      ultimately have "?rt1 ⊑ oip ?rt2 oip" by (rule rt_strictly_fresher_eqI)
      with ⟨dip = oip⟩ show ?thesis by simp qed
  qed

} note rreq_rrep_update = this

have "opaodv 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_invariant_sterms [OF aodv_wf rreq_sip [THEN weaken]]
  onl_invariant_sterms [OF aodv_wf rrep_sip [THEN weaken]]
  onl_invariant_sterms [OF aodv_wf rerr_sip [THEN weaken]]
  onl_invariant_sterms [OF aodv_wf oosn_rreq [THEN weaken]]
  onl_invariant_sterms [OF aodv_wf odsn_rrep [THEN weaken]]
  solve: basic update_0_unk invalidate rreq_rrep_update
  simp add: seqlsimp)
fix σ σ' p l
assume or: "(σ, p) ∈ oreachable (opaodv i) (?S i) (other quality_increases {i})"
  and "other quality_increases {i} 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 (opaodv 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 oaodv_trans aodv_trans]
otherwith_actionD
simp: seqlsimp)

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 (?nhip))"
and nsqns: "nsqn (rt (σ i)) dip ≤ nsqn (rt (?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 (?nhip)) dip"
by (rule nsqns)
also have "... ≤ sqn (rt (?nhip)) dip"
by (rule nsqn_sqn)
finally have "0 < sqn (rt (?nhip)) dip".

have "rt (σ i) ⊏ dip rt (σ' ?nhip)"
proof (cases "dip ∈ vD (rt (?nhip))")
assume "dip ∈ vD (rt (?nhip))"
with pre ⟨dip ∈ vD (rt (σ i))⟩ and ⟨?nhip ≠ dip⟩
have "rt (σ i) ⊏ dip rt (?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 (?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)
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!: onlD)

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} →)
  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

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

Given the state of a network σ, 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. r dip = Some (dsn, dsk, val, hops, ip')))"
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

end

3.11 Lift and transfer invariants to show loop freedom

theory C_Aodv_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 Γ AODV i) (seqp_sos Γ QMSG)"
  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}) (orecmsg (λσ m.
      msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases {i} →))

325
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_nhup_quality_increases])

show "opaodv i \models_A (otherwith (op=) {i})
(orecvmsg (λm. msg_fresh σ m ∧ msg_zhops m)),
other quality_increases {i} \rightarrow
"globala (λ(σ', σ), σ'). quality_increases (σ i) (σ' i))"

proof (rule ostep_invariant_weakenE [OF oquality_increases], simp_all)

fix t :: "(((nat ⇒ state) × (state, msg, pseq, pseq label) seqp), msg seq_action) transition"
assume "onll Γ_AODV (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' a
assume "otherwith (op=) {i}
(orecvmsg (λm. msg_fresh σ m ∧ msg_zhops m)) σ σ' a"
thus "otherwith quality_increases {i} (orecvmsg (λ_. req_rep_sn)) σ σ' a"
by - (erule weaken_otherwith, auto)

qed

lemma par_req_rrep_sn_quality_increases:
"opaodv i \models_A (λσ _. orecvmsg (λ_. req_rep_sn) σ, other (λ_ _. True) {i} \rightarrow
"globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"

proof -

have "opaodv i \models_A (λσ _. orecvmsg (λ_. req_rep_sn) σ, other (λ_ _. True) {i} \rightarrow
"globala (λ(σ, _, σ'), quality_increases (σ i) (σ' i))"
by (rule ostep_invariant_weakenE [OF olocal_quality_increases])
(auto dest!: onllD seqllD elim!: aodv_ex_labelE)

hence "opaodv i \models_A (λσ _. orecvmsg (λ_. req_rep_sn) σ, other (λ_ _. True) {i} \rightarrow
"globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
by (rule lift_step_into_qmsg_statelessassm) simp_all
thust thesis by rule auto

qed

lemma par_req_rep_snqns_sn_quality_increases:
"opaodv i \models_A (λσ _. orecvmsg (λ_. req_rep_sn) σ, other (λ_ _. True) {i} \rightarrow
"globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)"

proof -

have "opaodv i \models_A (λσ _. orecvmsg (λ_. req_rep_sn) σ, other (λ_ _. True) {i} \rightarrow
"globala (λ(σ, a, σ'), anycast (msg_fresh σ) a)"
by (rule ostep_invariant_weakenE [OF req_rep_snqns_sn_quality_increases])
(auto dest!: onllD seqllD elim!: aodv_ex_labelE)

hence "opaodv i \models_A (λσ _. orecvmsg (λ_. req_rep_sn) σ, other (λ_ _. True) {i} \rightarrow
"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 "opaodv i \models_A (λσ _. orecvmsg (λ_. req_rep_sn) σ, other (λ_ _. True) {i} \rightarrow
"globala (λ(σ, a, _). anycast msg_zhops a)"

proof -

from anycast_msg_zhops initiali_aodv oadv_trans aodv_trans

have "opaodv i \models_A (act TT, other (λ_ _. True) {i} \rightarrow
seqll i (onll Γ_AODV (λ_, a, _). anycast msg_zhops a))"
by (rule open_seq_step_invariant)

hence "opaodv i \models_A (λσ _. orecvmsg (λ_. req_rep_sn) σ, other (λ_ _. True) {i} \rightarrow
"globala (λ(σ, a, _). anycast msg_zhops a)"
by (rule lift_step_into_qmsg_statelessassm) simp_all
thus thesis by rule auto

qed
proof (rule ostep_invariant_weakenE)
fix t :: "\((\text{nat} \Rightarrow \text{state}) \times (\text{state}, \text{msg}, \text{seqp}, \text{seqp}) \text{ seqp}) \text{ seqp} \text{ label} \text{ seqp}) \text{ transition}\"
assume "seqll i (onll AODV (\(_, a, _). anycast msg_zhops a)) t"
thus "globala (\(_, a, _). anycast msg_zhops a) t"
by (cases t) (clarsimp dest!: seqllD onllD, metis aodv_ex_label)
qed simp_all

3.11.2 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 \text{onode_sos (oparp_sos i (oseqp_sos i) (seqp_sos \text{Gamma} \text{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 
  \(\langle i : \text{opaodv i} \mid \langle i \text{qmsg : R} \rangle \odot (= A (\lambda_\_. \text{orcremsg} (\lambda_. \text{rreq_rrep_sn}) \sigma, \text{other} (\lambda_. \text{True}) \{i\} \rightarrow)
  \text{globala} (\lambda(_, a, _). \text{cmsgmsg (msg_fresh (\sigma a))})\)
by (rule node_lift [OF par_nhop_quality_increases]) simp

lemma node_quality_increases:
  shows 
  \(\langle i : \text{opaodv i} \mid \langle i \text{qmsg : R} \rangle \odot (= A (\lambda_\_. \text{orcremsg} (\lambda_. \text{rreq_rrep_sn}) \sigma, \text{other} (\lambda_. \text{True}) \{i\} \rightarrow)
  \text{globala} (\lambda(_, a, _). \text{cmsgmsg (msg_fresh (\sigma a))})\)
by (rule node_lift_step_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp

lemma node_rreq_rrep_nsqn_fresh_any_step:
  shows 
  \(\langle i : \text{opaodv i} \mid \langle i \text{qmsg : R} \rangle \odot (= A (\lambda_\_. \text{orcremsg} (\lambda_. \text{rreq_rrep_sn}) \sigma, \text{other} (\lambda_. \text{True}) \{i\} \rightarrow)
  \text{globala} (\lambda(_, a, _). \text{cmsgmsg (msg_fresh (\sigma a))})\)
by (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_nsqn_fresh_any_step])

lemma node_anycast_msg_zhops:
  shows 
  \(\langle i : \text{opaodv i} \mid \langle i \text{qmsg : R} \rangle \odot (= A (\lambda_\_. \text{orcremsg} (\lambda_. \text{rreq_rrep_sn}) \sigma, \text{other} (\lambda_. \text{True}) \{i\} \rightarrow)
  \text{globala} (\lambda(_, a, _). \text{cmsgmsg (msg_zhops a)})\)
by (rule node_lift_anycast_statelessassm [OF par_anycast_msg_zhops])

lemma node_silent_change_only:
  shows 
  \(\langle i : \text{opaodv i} \mid \langle i \text{qmsg : R} \rangle \odot (= A (\lambda_\_. \text{orcremsg} (\lambda_. \text{rreq_rrep_sn}) \sigma, \text{other} (\lambda_. \text{True}) \{i\} \rightarrow)
  \text{globala} (\lambda(_, a, _). \text{cmsgmsg (msg_zhops a)})\)
proof (rule ostep_invariantI, simp (no_asm), rule impl)
  fix \(\sigma \cdot a \cdot (a', \cdot)\)
  assume or: "\((\sigma, \cdot) \in \text{oreachable} (\langle i : \text{opaodv i} \mid \langle i \text{qmsg : R} \rangle)\)
  (\lambda_\_. \text{orcremsg (l_\text{True}) \{i\} \rightarrow}
  \text{globala} (\lambda(_, a, _). \text{cmsgmsg (msg_zhops a)})\)
  and tr: "\((\sigma, \cdot), a, (a', \cdot) \in \text{trans} (\langle i : \text{opaodv i} \mid \langle i \text{qmsg : R} \rangle)\)
  and "a \neq \tau_n"
from or obtain p R where "\(\cdot = \text{NodeS } i \in p R\)"
by - (drule node_net_state, metis)
with \( \text{tr} \) have "\(((\sigma, \text{NodeS i p R}), a, (\sigma', \zeta')) \in \text{onode sos (oparp sos i (trans (opaodv i)) (trans qmsg)))" by simp 
thus "\( \sigma' = \sigma \)" using \( a \neq \tau \)
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_msg_nsqn_fresh_inc_sn [simp]:
assumes "\( \text{oarrivemsg (} \lambda \sigma m. \sigma \) \)
shows "\( \text{oarrivemsg (} \lambda \sigma m. \sigma \) \)
using assms by (cases m) auto

lemma opnet_nhop_quality_increases:
shows "\( \text{opnet (} \lambda i. \text{opaodv i) (} \langle i \rangle \) \)
proof (rule pnet_lift [OF node_nhop_quality_increases])
fix \( i R \)
have "\( \langle i : \text{opaodv i (} \langle i \rangle \) \)
proof (rule ostep_invariantI, simp (no_asm))
fix \( \sigma s a \)
assume or: "\( i \in \text{oreachable (} \lambda \sigma m. \sigma \) \)
and tr: "\( ((\sigma, s), a, (\sigma', s')) \in \text{trans (} \lambda i. \text{opaodv i (} \langle i \rangle \) \)
and am: "\( \text{castmsg (} \lambda \sigma m. \sigma \) \)
from or tr am have "\( \text{castmsg (} \lambda \sigma m. \sigma \) \)
by (auto dest!: ostep_invariantD [OF node_rreq_rrep_msg_nsqn_fresh_any_step])
moreover from or tr am have "\( \text{castmsg (} \lambda m. \sigma \) \)
by (auto dest!: ostep_invariantD [OF node_anycast_msg_zhops])
ultimately show "\( \text{castmsg (} \lambda m. \sigma \) \)
by (case_tac a) auto
qed

next
fix \( i R \)
show "\( \langle i : \text{opaodv i (} \langle i \rangle \) \)
by (rule ostep_invariant_weakenE [OF node_silent_change_only]) auto
next
fix \( i R \)
show "\( \langle i : \text{opaodv i (} \langle i \rangle \) \)
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 "onet_nhop_quality_increases: "oclosed (opnet (λi. opaodv i ⟨⟨i qmsg) 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 "_ => (l, ?U -> ?inv")
proof (rule inclosed_closed)
from opnet_nhop_quality_increases
show "opnet (λi. opaodv i ⟨⟨i qmsg) p
    | (otherwith (op=) (net_tree_ips p) inoclosed, ?U -> ?inv"
proof (rule oinvariant_weakenE)
fix σ σ ' :: "ip => state" and a :: "msg node_action"
assume "otherwith (op=) (net_tree_ips p) inoclosed σ σ '
a" otherwise 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

3.11.5 Transfer into the standard model

interpretation aodv_openproc: openproc paodv opaodv id
where "aodv_openproc.initmissing = initmissing"
proof -
  show "openproc paodv opaodv id"
proof unfold_locales
  fix i :: ip
  have "{(σ, ζ). (σ i, ζ) ∈ σ AODV i ∧ (∀j. j ≠ i → σ j ∈ fst ' σ AODV j)} ⊆ σ AODV,'" unfolding σ AODV_def σ AODV'_def
proof (rule equalityD1)
  show "∀p. {(σ i, ζ). (σ i, ζ) ∈ f i, p} ∧ (∀j. j ≠ i → σ j ∈ fst ' f j, p)} = f i, p)" by (rule set_eqI) auto
qed
thus "{ σ ζ s. s ∈ init (paodv i)
    ∧ (σ i, ζ) = id s
    ∧ (∀j. j ≠ i → σ j ∈ (fst o id) ' init (paodv j)) } ⊆ init (opaodv i)"
by simp
next
show "∀j. init (paodv 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 (paodv i)" then obtain q q' where "s = (σ i, q)"
and "s' = (σ' i, q')" and "(σ i, q), a, (σ' i, q') ∈ trans (paodv i)"
by (cases s, cases s') auto
from this(3) have 
\((\sigma, q, a, (\sigma', q')) \in \text{trans}(\text{opaodv } i)\)
by simp (rule open_seqp_action \[OF aodv_wf\])

with \(s = (\sigma, q)\) and \(s' = (\sigma', q')\)
show 
\((((\sigma, \text{snd}(\text{id } s)), a, (\sigma', \text{snd}(\text{id } s')))) \in \text{trans}(\text{opaodv } i)\)
by simp

qed

then interpret op: openc proc paodv opaodv id

have [simp]: 
\(\forall i. (\text{SOME } x. x \in (\text{fst } \text{id } s) \quad \text{init}(\text{paodv } i)) = \text{aodv_init } i\)
unfolding op\.AODV\_def by simp

hence 
\(\forall i. \text{openproc.initmissing paodv id } i = \text{initmissing } i\)
unfolding opq.initmissing_def opq.someinit_def initmissing_def
by (auto split: option.split)

thus
\(\text{openproc.initmissing paodv id } = \text{initmissing}\)
by auto

qed

interpretation aodv_openproc_par_qmsg: openproc_parq paodv opaodv id qmsg
where \(\text{aodv_openproc_par_qmsg.netglobal } = \text{netglobal}\)
and \(\text{aodv_openproc_par_qmsg.initmissing } = \text{initmissing}\)
proof
-show "openproc_parq paodv opaodv id qmsg"
by (unfold_locales) simp
then interpret opq: openproc_parq paodv opaodv id qmsg

have im: 
\(\forall P \sigma. \text{openproc.netglobal } (\lambda i. \text{paodv } i \langle\langle qmsg \rangle\rangle \langle\langle \lambda (p, q). (\text{fst}(\text{id } p), \text{snd}(\text{id } p), q)\rangle\rangle P \sigma = \text{netglobal } P \sigma\)
unfolding opq.netglobal_def netglobal_def opq.initmissing_def initmissing_def opq.someinit_def
unfolding op\.AODV\_def op\.QMSG\_def
by (clarsimp cong: option.case_cong
simp del: One_nat_def
simp add: fst_initmissing_netgmap_default_aodv_init_netlift [symmetric, unfolded initmissing_def])
thus
\(\text{openproc.netglobal } (\lambda i. \text{paodv } i \langle\langle qmsg \rangle\rangle \langle\langle \lambda (p, q). (\text{fst}(\text{id } p), \text{snd}(\text{id } p), q)\rangle\rangle = \text{netglobal}\)
by auto

qed

lemma net_nhop_quality_increases:
assumes "\text{wf_net_tree } n"
shows "\text{closed } (\text{pnet } (\lambda i. \text{paodv } i \langle\langle qmsg \rangle\rangle n) \models \text{netglobal}\)\)
\((\lambda \sigma. \forall i \text{ dip. let nhip } = \text{the (nhop } \text{rt } (\sigma \ i)) \text{ dip } \in \text{V } \text{dip } \in \text{V } \text{dip } \in \text{V}\) \wedge \text{nhip } \neq \text{dip}\)
\(\rightarrow (\text{rt } (\sigma \ i)) \text{ dip } \rightarrow (\text{rt } (\sigma \ nhip))\)"
(is "\text{_ } \models \text{netglobal } (\lambda \sigma. \forall i. \text{?inv } \sigma \ i)"")
proof -
from \(\text{wf_net_tree } n\)
have proto: "\text{closed } (\text{pnet } (\lambda i. \text{paodv } i \langle\langle qmsg \rangle\rangle n) \models \text{netglobal}\)\)
\((\lambda \sigma. \forall i \text{ dip. let nhip } = \text{the (nhop } \text{rt } (\sigma \ i)) \text{ dip } \in \text{V } \text{dip } \in \text{V } \text{dip } \in \text{V}\) \wedge \text{nhip } \neq \text{dip}\)
\(\rightarrow (\text{rt } (\sigma \ i)) \text{ dip } \rightarrow (\text{rt } (\sigma \ nhip))\)"
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
\text{fst.initmissing.netgmap.pair_fst}, rule allI)
fix \(\sigma \ i\)
assume sr: "\(\sigma \in \text{reachable } (\text{closed } (\text{pnet } (\lambda i. \text{paodv } i \langle\langle qmsg \rangle\rangle n)) \text{ TT}\)"
hence "\(\forall i \in \text{net_tree_ips } n. \text{?inv } (\text{fst } (\text{initmissing } (\text{netgmap } \text{fst } \sigma))) i\)"
by -(drule invariantD [OF proto],

330
simp only: aodv_openproc_par_qmsg.netglobalsimp
  fst_initmissing_netgmap_pair_fst)
thus "?inv (fst (initmissing (netgmap fst σ))) i"
proof (cases "i \in net_tree_ips n")
  assume "i \notin net_tree_ips n"
  from sr have "σ \in reachable (pnet (λi. paodv i \langle qmsg \rangle n) TT)" ..
  hence "net_ips σ = net_tree_ips n" ..
  with (i \notin net_tree_ips n) have "i \notin net_ips σ" by simp
  hence "(fst (initmissing (netgmap fst σ))) i = aodv_init i"
    by simp
  thus ?thesis by simp
qed metis
qed

3.11.6 Loop freedom of AODV

theorem aodv_loop_freedom:
  assumes "wf_net_tree n"
  shows "closed (pnet (λi. paodv i \langle qmsg \rangle n) \parallel netglobal (λσ. \forall dip. irrefl ((rt_graph σ dip)⁺)))"
using assms by (rule aodv_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 ⇒ sqn"
  where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ 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 ≠ Suc 0"
  unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 1"
  by simp

4.1.2 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 × k × f × nat × ip × ip set"

definition proj2 :: "r ⇒ sqn" ("\pi_2")
where \( \pi_2 \equiv \lambda (dsn, _, _, _, _, _). dsn \)

**Definition**

\[ \text{proj3 :: } \text{"r } \Rightarrow \text{ k ("\pi_3"\)}} \]

where \( \pi_3 \equiv \lambda (dsn, dsk, _, _, _, _). dsk \)

**Definition**

\[ \text{proj4 :: } \text{"r } \Rightarrow \text{ f ("\pi_4"\)}} \]

where \( \pi_4 \equiv \lambda (dsn, dsk, flag, _, _, _). flag \)

**Definition**

\[ \text{proj5 :: } \text{"r } \Rightarrow \text{ nat ("\pi_5"\)}} \]

where \( \pi_5 \equiv \lambda (dsn, dsk, flag, hops, _, _). hops \)

**Definition**

\[ \text{proj6 :: } \text{"r } \Rightarrow \text{ ip ("\pi_6"\)}} \]

where \( \pi_6 \equiv \lambda (dsn, dsk, flag, hops, nhip, _. pre). nhip \)

**Definition**

\[ \text{proj7 :: } \text{"r } \Rightarrow \text{ ip set ("\pi_7"\)}} \]

where \( \pi_7 \equiv \lambda (dsn, dsk, flag, hops, nhip, pre). pre \)

**Lemma**

\[ \text{projs [simp]: } \pi_2 (dsn, dsk, flag, hops, nhip, pre) = dsn \]

\[ \pi_3 (dsn, dsk, flag, hops, nhip, pre) = dsk \]

\[ \pi_4 (dsn, dsk, flag, hops, nhip, pre) = flag \]

\[ \pi_5 (dsn, dsk, flag, hops, nhip, pre) = hops \]

\[ \pi_6 (dsn, dsk, flag, hops, nhip, pre) = nhip \]

\[ \pi_7 (dsn, dsk, flag, hops, nhip, pre) = pre \]

by (clarsimp simp: proj2_def proj3_def proj4_def proj5_def proj6_def proj7_def)+

**Lemma**

\[ \text{proj3_pred [intro]: } \forall P \text{ kno; } P \text{ unk } \Rightarrow P (\pi_3 x) \]

by (rule k.induct)

**Lemma**

\[ \text{proj4_pred [intro]: } \forall P \text{ val; } P \text{ inv } \Rightarrow P (\pi_4 x) \]

by (rule f.induct)

**Lemma**

\[ \text{proj6_pair_snd [simp]: } \text{ fixes dsn' r } \]

shows \( \pi_6 (dsn', snd (r)) = \pi_6 (r) \)

by (cases r) simp

### 4.1.3 Routing Tables

Routing tables map ip addresses to route entries.

**Type Synonym**

\( \text{rt = "ip } \Rightarrow \text{ r"} \)

**Syntax**

\( "\_Sigma_route" :: "\text{rt } \Rightarrow \text{ ip } \Rightarrow \text{ r} ("\sigma_{route'}(\_\_')")" \)

**Translations**

\( "\sigma_{route}(rt, dip)" = "\text{rt dip}" \)

**Definition**

\( \text{sqn :: } \text{rt } \Rightarrow \text{ ip } \Rightarrow \text{ sqn} \)

where \( \text{sqn rt dip } \equiv \text{ case } \sigma_{route}(rt, dip) \text{ of Some } r \Rightarrow \pi_2 (r) \mid \text{None } \Rightarrow 0 \)

**Definition**

\( \text{sqnf :: } \text{rt } \Rightarrow \text{ ip } \Rightarrow \text{ k} \)

where \( \text{sqnf rt dip } \equiv \text{ case } \sigma_{route}(rt, dip) \text{ of Some } r \Rightarrow \pi_3 (r) \mid \text{None } \Rightarrow \text{ unk} \)

**Abbreviation**

\( \text{flag :: } \text{rt } \Rightarrow \text{ ip } \Rightarrow \text{ f} \)

where \( \text{flag rt dip } \equiv \text{ map_option } \pi_4 (\sigma_{route}(rt, dip)) \)

**Abbreviation**

\( \text{dhops :: } \text{rt } \Rightarrow \text{ ip } \Rightarrow \text{ nat} \)

where \( \text{dhops rt dip } \equiv \text{ map_option } \pi_5 (\sigma_{route}(rt, dip)) \)

**Abbreviation**

\( \text{nhop :: } \text{rt } \Rightarrow \text{ ip } \Rightarrow \text{ ip} \)

where \( \text{nhop rt dip } \equiv \text{ map_option } \pi_6 (\sigma_{route}(rt, dip)) \)

333
abbreviation \texttt{precs :: "rt \Rightarrow ip \rightarrow ip\ set"}
\hspace{1cm} where \texttt{precs rt dip \equiv map\_option \pi_7 (\sigma\ \text{route}(rt, dip))} \\

definition \texttt{vD :: "rt \Rightarrow ip\ set"}
\hspace{1cm} where \texttt{vD rt \equiv \{dip. flag rt dip = Some val\}} \\

definition \texttt{iD :: "rt \Rightarrow ip\ set"}
\hspace{1cm} where \texttt{iD rt \equiv \{dip. flag rt dip = Some inv\}} \\

definition \texttt{kD :: "rt \Rightarrow ip\ set"}
\hspace{1cm} where \texttt{kD rt \equiv \{dip. rt dip \neq None\}} \\

lemma \texttt{kD\_is\ vD\_and\ iD}: \texttt{"kD rt = vD rt \cup iD rt"}
\hspace{1cm} unfolding \texttt{kD\_def vD\_def iD\_def by auto} \\

lemma \texttt{vD\_iD\_gives\ kD} [simp]:
\hspace{1cm} \"\\texttt{\forall ip rt. ip \in vD rt \implies ip \in kD rt}\" \\
\hspace{1cm} \"\\texttt{\forall ip rt. ip \in iD rt \implies ip \in kD rt}\"
\hspace{1cm} unfolding \texttt{kD\_is\ vD\_and\ iD} by \texttt{simp\_all}

lemma \texttt{kD\_Some} [dest]:
\hspace{1cm} \texttt{fixes dip rt}
\hspace{1cm} \texttt{assumes \"dip \in kD rt\"}
\hspace{1cm} \texttt{shows \"\exists dsn dsk flag hops nhip pre.}
\hspace{2cm} \texttt{\sigma\ \text{route}(rt, dip) = Some (dsn, dsk, flag, hops, nhip, pre)\"}
\hspace{1cm} using \texttt{assms unfolding kD\_def by simp} \\

lemma \texttt{kD\_None} [dest]:
\hspace{1cm} \texttt{fixes dip rt}
\hspace{1cm} \texttt{assumes \"dip \notin kD rt\"}
\hspace{1cm} \texttt{shows \"\sigma\ \text{route}(rt, dip) = None\"}
\hspace{1cm} using \texttt{assms unfolding kD\_def}
\hspace{1cm} by \texttt{(metis (mono_tags) mem\_Collect\_eq)} \\

lemma \texttt{vD\_Some} [dest]:
\hspace{1cm} \texttt{fixes dip rt}
\hspace{1cm} \texttt{assumes \"dip \in vD rt\"}
\hspace{1cm} \texttt{shows \"\exists dsn dsk hops nhip pre.}
\hspace{2cm} \texttt{\sigma\ \text{route}(rt, dip) = Some (dsn, dsk, val, hops, nhip, pre)\"}
\hspace{1cm} using \texttt{assms unfolding vD\_def by simp} \\

lemma \texttt{vD\_empty} [simp]: \texttt{"vD Map.empty = {}"}
\hspace{1cm} unfolding \texttt{vD\_def by simp} \\

lemma \texttt{iD\_Some} [dest]:
\hspace{1cm} \texttt{fixes dip rt}
\hspace{1cm} \texttt{assumes \"dip \in iD rt\"}
\hspace{1cm} \texttt{shows \"\exists dsn dsk hops nhip pre.}
\hspace{2cm} \texttt{\sigma\ \text{route}(rt, dip) = Some (dsn, dsk, inv, hops, nhip, pre)\"}
\hspace{1cm} using \texttt{assms unfolding iD\_def by simp} \\

lemma \texttt{val\_is\ vD} [elim]:
\hspace{1cm} \texttt{fixes ip rt}
\hspace{1cm} \texttt{assumes \"ip \in kD(rt)\"}
\hspace{1cm} \texttt{and \"the (flag rt ip) = val\"}
\hspace{1cm} \texttt{shows \"ip \in vD(rt)\"}
\hspace{1cm} using \texttt{assms unfolding vD\_def by auto} \\

lemma \texttt{inv\_is\ iD} [elim]:
\hspace{1cm} \texttt{fixes ip rt}
\hspace{1cm} \texttt{assumes \"ip \in kD(rt)\"}
\hspace{1cm} \texttt{and \"the (flag rt ip) = inv\"}
\hspace{1cm} \texttt{shows \"ip \in iD(rt)\"}
\hspace{1cm} using \texttt{assms unfolding iD\_def by auto}
lemma $iD$ _flag_is_inv [elim, simp]:
  fixes ip rt
  assumes "ip$\in$iD(rt)"
  shows "the (flag rt ip) = inv"
proof -
  from (ip$\in$iD(rt)) have "ip$\in$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$\in$kD(rt)"
  and "ip /$\in$vD(rt)"
  shows "ip$\in$iD(rt)"
proof -
  from (ip$\in$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 /$\in$vD(rt)) have "f \neq val"
    proof (rule contrapos_nn)
      assume "f = val"
      with rtip have "the (flag rt ip) = val" by simp
      with (ip$\in$kD(rt)) show "ip$\in$vD(rt)" ..
    qed
  with rtip have "the (flag rt ip) = inv" by simp
  with (ip$\in$kD(rt)) show "ip$\in$iD(rt)" ..
  qed

lemma vD_or_iD [elim]:
  fixes ip rt
  assumes "ip$\in$kD(rt)"
  and "ip$\in$vD(rt) = P rt ip"
  and "ip$\in$iD(rt) = P rt ip"
  shows "P rt ip"
proof -
  from (ip$\in$kD(rt)) have "ip$\in$vD(rt) $\cup$ iD(rt)"
    by (simp add: kD_is_vD_and_iD)
  thus ?thesis by (auto elim: assms(2-3))
  qed

lemma proj_5_eq_dhops: "$\forall$ dip rt. dip$\in$kD(rt) $\Rightarrow$ $\pi$5(the (rt dip)) = the (dhops rt dip)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj_4_eq_flag: "$\forall$ dip rt. dip$\in$kD(rt) $\Rightarrow$ $\pi$4(the (rt dip)) = the (flag rt dip)"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma proj_2_eq_sqn: "$\forall$ dip rt. dip$\in$kD(rt) $\Rightarrow$ $\pi$2(the (rt dip)) = sqn rt dip"
  unfolding sqn_def by (drule kD_Some) clarsimp

lemma kD_sqnf_is_proj3 [simp]:
  "$\forall$ ip rt. ip$\in$kD(rt) $\Rightarrow$ sqnf rt ip = $\pi$3(the (rt ip))"
  unfolding sqnf_def by auto

lemma vD_flag_val [simp]:
  "$\forall$ dip rt. dip$\in$vD rt $\Rightarrow$ the (flag rt dip) = val"
  unfolding vD_def by clarsimp

lemma kD_update [simp]:
  "$\forall$ rt nip v. kD (rt nip $\mapsto$ 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

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 ⇒ 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) ∪ npre"
  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 set ⇒ rt"
  where "addpreRT rt dip npre ≡ map_option (λs. rt (dip ⊢ addpre s npre)) (σ_route rt dip)"
lemma \text{snd\_addpre} [simp]:
\[ ((dsn, dsn', v, pre)) = addpre (dsn, v) pre \]
unfolding addpre\_def by clarsimp

lemma \text{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 \text{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 \text{proj5\_addpreRT} [simp]:
\[ \forall rt dip ip npre. dip\in kD (rt) \implies \pi_5 (the (the (addpreRT rt dip npre) ip)) = \pi_5 (the (rt ip)) \]
unfolding addpreRT\_def by auto

lemma \text{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 \text{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 \text{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 \text{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 \text{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 \text{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]:
  "∀ip dip. ip ∈ kD rt =⇒ sqnf (the (addpreRT rt dip npre)) dip = sqnf rt 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 ≡ π_4(r) = val ∧
        (π_2(r) = 0) = (π_3(r) = unk) ∧
        (π_3(r) = unk −→ π_5(r) = 1)"

lemma update_arg_wf_gives_cases:
  "∀r. update_arg_wf r =⇒ (π_2(r) = 0) = (π_3(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 simp

lemma update_arg_wf_tuples' [elim]:
  "∀n hops nhip pre. Suc 0 ≤ n =⇒ 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 cl1: "∀nhip pre. P (0, unk, val, Suc 0, nhip, pre)"
  and cl2: "∀dsn hops nhip pre. dsn > 0 =⇒ 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 \(wf3: \"dsk = unk \rightarrow (hops = 1)"\)

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

definition \(update\ :: \"rt \Rightarrow ip \Rightarrow r \Rightarrow rt\)\)
where
\(\"update\ \(rt\ \(ip\ \(r\) ≡\)
\case \(σ\_route\(\(rt, ip\) of
\(\\)None ⇒ \(rt\ \(ip\ ⇒ \(r)\)
\| \(\)Some \(s \⇒\)
\(\)if \(\pi_5(s) < \pi_5(r)\) then \(rt\ \(ip\ ⇒ \(addpre\ \(r\ \(\pi_7(s)\)))\)
\(\)else if \(\pi_5(s) = \pi_5(r)\) \∧ \(\pi_5(s) > \pi_5(r)\) \∨ \(\pi_4(s) = \text{inv}\)
\(\)then \(rt\ \(ip\ ⇒ \(addpre\ \(r\ \(\pi_7(s)\)))\)
\(\)else if \(\pi_3(r) = \text{unk}\)
\(\)then \(rt\ \(ip\ ⇒ \(\pi_3(s), \text{snd} (addpre\ \(r\ \(\pi_7(s)\)))\))\)
\(\)else if \(\pi_3(r) = \text{kno}\)
\(\)then \(rt\ \(ip\ ⇒ \(\pi_3(s), \text{addpre}\ \(s\ \(\pi_7(r)\))\))\)
\(\)else \(rt\ \(ip\ ⇒ \(\pi_3(s)\))\)"
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 \( \langle \text{sqn rt ip} = \pi_2(r) \rangle \) and \( \langle \text{the (dhops rt ip)} > \pi_5(r) \rangle \)

show "update rt ip r = rt (ip \mapsto \text{nr})"

unfolding update_def nr_def s_def by auto

next

assume "ip \in kD(rt)"

and "sqn rt ip = \( \pi_3(r) \)"

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 \( \langle \text{sqn rt ip} = \pi_3(r) \rangle \) and \( \langle \text{flag rt ip} = Some \text{ \textit{inv}} \rangle \)

show "update rt ip r = rt (ip \mapsto \text{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 \( \langle \pi_2(r) = 0 \rangle \) = \( \langle \pi_3(r) = \text{unk} \) and \( \langle \pi_3(r) = \text{unk} \)

show "update rt ip r = rt (ip \mapsto \text{nr})"

unfolding update_def nr'_def nr_def s_def by (cases r) simp

next

assume "ip \in kD(rt)"

and \textit{otherassms}: "sqn rt ip \geq \( \pi_2(r) \)"

"\( \pi_3(r) = \text{\textit{kno}} \)"

"sqn rt ip = \( \pi_2(r) \) \Rightarrow \text{the (dhops rt ip)} \leq \( \pi_5(r) \) \land \text{the (flag rt ip)} = \text{\textit{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 \textit{otherassms} show "update rt ip r = rt (ip \mapsto \text{ns})"

unfolding update_def ns_def s_def by auto

qed

lemma **update_cases [elim]**:

assumes "(\( \pi_3(r) = 0 \) = (\( \pi_3(r) = \text{unk} \))"

and \textbf{c1}: "[\[ \text{ip} \notin kD(rt) \] \Rightarrow P (rt (ip \mapsto r))"

and \textbf{c2}: "[\[ \text{ip} \in kD(rt); sqn rt ip < \( \pi_2(r) \) \]

\Rightarrow P (rt (ip \mapsto \text{addpre r (\( \pi_7(\text{route}(rt, ip)))))}\))"

and \textbf{c3}: "[\[ \text{ip} \in kD(rt); sqn rt ip = \( \pi_2(r) \); the (dhops rt ip) > \( \pi_5(r) \) \]

\Rightarrow P (rt (ip \mapsto \text{addpre r (\( \pi_7(\text{route}(rt, ip)))))}\))"

and \textbf{c4}: "[\[ \text{ip} \in kD(rt); sqn rt ip = \( \pi_2(r) \); \text{the (flag rt ip)} = \text{\textit{inv}} \]

\Rightarrow P (rt (ip \mapsto \text{addpre r (\( \pi_7(\text{route}(rt, ip)))))}\))"

and \textbf{c5}: "[\[ \text{ip} \in kD(rt); \pi_3(r) = \text{unk} \]

\Rightarrow P (rt (ip \mapsto (\( \pi_7(\text{route}(rt, ip)), \pi_3(r),

\pi_4(r), \pi_5(r), \pi_6(r), \pi_7(\text{addpre r (\( \pi_7(\text{route}(rt, ip)))))}\))")

and \textbf{c6}: "[\[ \text{ip} \in kD(rt); sqn rt ip \geq \( \pi_2(r) \); \pi_3(r) = \text{\textit{kno}};\)

sqn rt ip = \( \pi_2(r) \) \Rightarrow \text{the (dhops rt ip)} \leq \( \pi_5(r) \) \land \text{the (flag rt ip)} = \text{\textit{val}}\]

\Rightarrow P (rt (ip \mapsto \text{addpre (\( \pi_7(\text{route}(rt, ip)))) (\pi_7(r))))\))"

shows "(P (update rt ip r))"

proof (cases "ip \in kD(rt)"

assume "\text{ip} \notin kD(rt)"

with \textbf{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 σ route (rt, ip)))))"
and c3: "[sqn rt ip = π2 r; the (dhops rt ip) > π5 r]
  =⇒ P (rt (ip ↦ addpre r (π7 the σ route (rt, ip)))))"
and c4: "[sqn rt ip = π2 r; the (flag rt ip) = inv]
  =⇒ P (rt (ip ↦ addpre r (π7 the σ route (rt, ip)))))"
and c5: "π3 r = unk =⇒ P (rt (ip ↦ (π2 (the σ route (rt, ip)), π3 r), π4 r, π5 r, π6 r, π7 (addpre r (π7 (the σ route (rt, ip)))))
  =⇒ P (rt (ip ↦ addpre (the σ route (rt, ip)) (π7 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 ↦ addpre (the σ route (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 σ route (rt, ip)), π3 r, π4 r, π5 r, π6 r, π7 (addpre r (π7 (the (rt ip)))))"
  =⇒ P (rt (ip ↦ addpre (the σ route (rt, ip)) (π7 r))))"
  =⇒ P (rt (ip ↦ addpre (the σ route (rt, ip)) (π7 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 ↦ addpre r (π7 (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 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

lemma sqn_if_updated:
  fixes rip v rt ip
shows "sqn (λ x. 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 dan dsk flag hops nip pre
  assumes "ip ≠ dip"
  shows "nhop (update rt dip (dsn, dsk, flag, hops, nip, pre)) ip = nhop rt ip"
  using assms unfolding update_def
  by (clarsimp split: option.split)

lemma dhops_update_another [simp]:
  fixes dip ip rt dan dsk flag hops nip pre
  assumes "ip ≠ dip"
  shows "dhops (update rt dip (dsn, dsk, flag, hops, nip, 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 nip pre. sqn (rt(ip ⧵→ v)) ip = π₂(v)"
  unfolding sqn_def
  by simp

lemma dhops_update_changed [simp]:
  fixes rt dip osn hops nip
  assumes "rt ≠ update rt dip (osn, kno, val, hops, nip, {})"
  shows "the (dhops (update rt dip (osn, kno, val, hops, nip, {})) 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 dsn hops nip. the (nhop (update rt dip (dsn, unk, val, hops, nip, pre)) 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 dip 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 flg hops sip.
  rt ≠ update rt dip (dsn, kno, flg, hops, sip, {})
  ⇒ sqnf (update rt dip (dsn, kno, 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 dsn hops nipre. 1 ≤ dsn ⇒ 1 ≤ (sqn (update rt dip (dsn, kno, val, hops, ip, nipre)) dip)"
  unfolding update_def
  by (clarsimp split: option.splits)

lemma proj3_update [simp]: "∀ rt dip dsn dsk flg hops sip.

343
\[ rt \neq \text{update } rt \text{ dip (dsn, dsk, flg, hops, sip, {})} \]
\[ \Rightarrow \pi_3(\text{the } (\text{update } rt \text{ dip (dsn, dsk, flg, hops, sip, {})} \text{ dip})) = \text{dsk} \]
unfolding update_def sqnf_def
by (clarsimp split: option.splits split_if_asm) auto

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

lemma flag_update [simp]: "\( \forall rt \text{ dip dsn flg hops sip.} \)
\[ rt \neq \text{update } rt \text{ dip (dsn, kno, flg, hops, sip, {})} \]
\[ \Rightarrow \text{the } (\text{flag } (\text{update } rt \text{ dip (dsn, kno, flg, hops, sip, {})} \text{ dip})) = \text{flg} \]
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 \in 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 = \text{update } rt \text{ dip (dsn, dsk, flag, hops, nhip, pre)}"
shows "dip \in kD(rt)"
proof -
  have "dip \in kD(update rt dip (dsn, dsk, flag, hops, nhip, pre))" by simp
  with assms show ?thesis by simp
qed

lemma nhop_update [simp]: "\( \forall rt \text{ dip dsn dsk flg hops sip.} \)
\[ rt \neq \text{update } rt \text{ dip (dsn, dsk, flg, hops, sip, {})} \]
\[ \Rightarrow \text{the } (\text{nhop } (\text{update } rt \text{ dip (dsn, dsk, flg, hops, sip, {})} \text{ dip})) = \text{sip} \]
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 \neq dip"
shows "sqn (update rt dip (dsn, dsk, flag, hops, nhip, pre)) ip = sqn rt ip"
using assms unfolding update_def sqnf_def
by (clarsimp split: option.splits) auto

lemma sqnf_update_another [simp]:
fixes dip ip rt dsn dsk flag hops nhip pre
assumes "ip \neq 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]:
"\( \forall dip \text{ rt dip' dsn dsk hops nhip pre.} \)
\[ dip \in vD(update rt dip' (dsn, dsk, val, hops, nhip, pre)) \Rightarrow (dip \in vD(rt) \lor dip=dip') \]
unfolding update_def vD_def by (clarsimp split: option.split_asm split_if_asm)

Invalidating route entries

definition invalidate :: "rt \Rightarrow (ip \Rightarrow sqn) \Rightarrow rt"
where "invalidate rt dests \equiv
\lambda ip. \text{case } (rt ip, dests ip) \text{ of}
(\text{None, _}) \Rightarrow \text{None} \Rightarrow \text{rt}"

344
lemma proj3_invalidate [simp]:
"\(\forall dip. \pi_3\left(\text{the}\ ((\text{invalidate} rt dests) dip)\right) = \pi_3\left(\text{the}\ (rt dip)\right)\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj5_invalidate [simp]:
"\(\forall dip. \pi_5\left(\text{the}\ ((\text{invalidate} rt dests) dip)\right) = \pi_5\left(\text{the}\ (rt dip)\right)\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj6_invalidate [simp]:
"\(\forall dip. \pi_6\left(\text{the}\ ((\text{invalidate} rt dests) dip)\right) = \pi_6\left(\text{the}\ (rt dip)\right)\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj7_invalidate [simp]:
"\(\forall dip. \pi_7\left(\text{the}\ ((\text{invalidate} rt dests) dip)\right) = \pi_7\left(\text{the}\ (rt dip)\right)\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_kD_inv [simp]:
"\(\forall rt dests. kD (\text{invalidate} rt dests) = kD rt\)"
unfolding invalidate_def kD_def
by (simp split: option.split)

lemma invalidate_sqn:
fixes rt dip dests
assumes "\(\forall rsn. \text{dests dip} = \text{Some}\ rsn \rightarrow sqn rt dip \leq rsn\)"
shows "\(sqn rt dip \leq sqn (\text{invalidate}\ rt dests) dip\)"
proof (cases "dip \notin kD(\text{rt})")
  assume "\(- dip \notin kD(\text{rt})\)"
  hence "dip \in kD(\text{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 \leq sqn (\text{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 "\(\text{dests ipa} = \text{Some}\ rsn\)"
  and "ipa \in kD(\text{rt})"
shows "\(sqn (\text{invalidate}\ rt dests) ipa = rsn\)"
unfolding invalidate_def sqn_def
using assms(1) assms(2) [THEN kD_Some]
by clarsimp

lemma dhops_invalidate [simp]:
"\(\forall dip. \text{the}\ ((\text{dhops}\ \text{invalidate} rt dests) dip) = \text{the}\ ((\text{dhops}\ rt dip))\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma sqnf_invalidate [simp]:
"\(\forall dip. \text{sqnf}\ ((\text{invalidate}\ rt \xi)\ (\text{dests}\ \xi)) dip = \text{sqnf}\ (rt \xi) dip\)"
unfolding sqnf_def invalidate_def by (clarsimp split: option.split)

lemma nhop_invalidate [simp]:
"\(\forall dip. \text{the}\ ((\text{nhop}\ (\text{invalidate}\ rt \xi)\ (\text{dests}\ \xi)) dip) = \text{the}\ ((\text{nhop}\ (rt \xi)) dip)\)"
unfolding invalidate_def by (clarsimp split: option.split)

lemma invalidate_other [simp]:
fixes rt dests dip
assumes "dip \notin \text{dom}(\text{dests})"
shows "\(\text{invalidate}\ rt dests dip = rt dip\)"
using assms unfolding invalidate_def
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:
  "\(\forall \) dip rt dests. dip ∈ vD(invalidate rt dests) \implies dip ∈ vD(rt) \land 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 \Rightarrow \pi_2(the (rt dip)) | Some rsn \Rightarrow rsn)
  \land dsk = \pi_3(the (rt dip))
  \land flag = (if dests dip = None then \pi_4(the (rt dip)) else inv)
  \land hops = \pi_5(the (rt dip))
  \land nhip = \pi_6(the (rt dip))
  \land pre = \pi_7(the (rt dip))"
  using assms unfolding invalidate_def
  by (cases "rt dip", clarsimp, cases "dests dip") auto

lemma proj3_inv: 
  "\(\forall \) dip rt dests. dip ∈ 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 ∈ 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" ("\(\sigma_{\text{queue}}(\_, \_\)"")
  where "\(\sigma_{\text{queue}}(\text{store}, dip) \equiv \text{case store dip of None ⇒ } [] \mid \text{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]))"
lemma \( qD_{\text{add}} [\text{simp}] \):
fixes \( d \) dip store
shows "\( qD(\text{add} \ d \ dip \ store) = \text{insert} \ dip \ (qD \ store) \)"
unfolding add_def Let_def qD_def
by (clarsimp split: option.split)

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

definition sigma_p_flag :: "\text{store} \Rightarrow \ \text{ip} \Rightarrow \ p"
where "\sigma_p-flag(store, dip) \equiv \text{map_option} \ \text{fst} \ (\text{store} \ dip)"

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

definition setRRF :: "\text{store} \Rightarrow \ \text{ip} \Rightarrow \ \text{store}"
where "setRRF \ store \ dests \equiv \ \lambda \ dip. \ \text{if} \ \text{dests} \ dip = \text{None} \ \text{then} \ \text{store} \ dip \ \text{else} \ \text{map_option} (\lambda(_, q). \ \text{req, q}) \ (\text{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. \( \text{nhop} \) is partial, thus a ‘the’ is needed, similarly for \( \text{dhops} \) and \( \text{addpreRT} \).
2. \( \text{precs} \) is partial.
3. \( \sigma_p-flag(store, dip) \) is partial.
4. The routing table (\( rt \)) is modelled as a map (\( \text{ip} \Rightarrow \ text{r option} \)) 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 \( \text{invalidate} \), and \( \text{store} \).
The \textit{msg} type models the different messages used within AODV. The instantiation as a \textit{msg} is a technicality due to the special treatment of \textit{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.

\textbf{definition} \textit{rreq} :: \textit{nat} \times \textit{rreqid} \times \textit{ip} \times \textit{sqn} \times \textit{k} \times \textit{ip} \times \textit{sqn} \times \textit{ip} \times \textit{bool} \Rightarrow \textit{msg}
where \textit{rreq} \equiv \lambda (\text{hops, rreqid, dip, dsn, dsk, oip, osn, sip, handled}).
\textit{Rreq} \text{hops} \text{rreqid} \text{dsn} \text{dsk} \text{oip} \text{osn} \text{sip} \text{handled}\

\textbf{lemma} \textit{rreq_simp} [simp]:
\textit{rreq}(\text{hops, rreqid, dip, dsn, dsk, oip, osn, sip, handled}) = \textit{Rreq} \text{hops} \text{rreqid} \text{dsn} \text{dsk} \text{oip} \text{osn} \text{sip} \text{handled}\
\text{unfolding} \textit{rreq_def} \text{by simp}\

\textbf{definition} \textit{rrep} :: \textit{nat} \times \textit{ip} \times \textit{sqn} \times \textit{ip} \times \textit{ip} \Rightarrow \textit{msg}
where \textit{rrep} \equiv \lambda (\text{hops, dip, dsn, oip, sip}). \textit{Rrep} \text{hops} \text{dsn} \text{oip} \text{sip}\

\textbf{lemma} \textit{rrep_simp} [simp]:
\textit{rrep}(\text{hops, dip, dsn, oip, sip}) = \textit{Rrep} \text{hops} \text{dsn} \text{oip} \text{sip}\
\text{unfolding} \textit{rrep_def} \text{by simp}\

\textbf{definition} \textit{rerr} :: \textit{(ip} \rightarrow \textit{sqn}) \times \textit{ip} \Rightarrow \textit{msg}
where \textit{rerr} \equiv \lambda (\text{dests, sip}). \textit{Rerr} \text{dests} \text{sip}\

\textbf{lemma} \textit{rerr_simp} [simp]:
\textit{rerr}(\text{dests, sip}) = \textit{Rerr} \text{dests} \text{sip}\
\text{unfolding} \textit{rerr_def} \text{by simp}\

\textbf{lemma} \textit{not_eq_newpkt_rreq} [simp]: \neg \textit{eq_newpkt} (\textit{Rreq} \text{hops} \text{rreqid} \text{dsn} \text{dsk} \text{oip} \text{osn} \text{sip} \text{handled})\
\text{unfolding} \textit{eq_newpkt_def} \text{by simp}\

\textbf{lemma} \textit{not_eq_newpkt_rrep} [simp]: \neg \textit{eq_newpkt} (\textit{Rrep} \text{hops} \text{dsn} \text{oip} \text{sip})\
\text{unfolding} \textit{eq_newpkt_def} \text{by simp}\

\textbf{lemma} \textit{not_eq_newpkt_rerr} [simp]: \neg \textit{eq_newpkt} (\textit{Rerr} \text{dests} \text{sip})\
\text{unfolding} \textit{eq_newpkt_def} \text{by simp}\

\textbf{lemma} \textit{not_eq_newpkt_pkt} [simp]: \neg \textit{eq_newpkt} (\textit{Pkt} \text{d} \text{dip} \text{sip})\
\text{unfolding} \textit{eq_newpkt_def} \text{by simp}\

\textbf{definition} \textit{pkt} :: \textit{data} \times \textit{ip} \times \textit{ip} \Rightarrow \textit{msg}
where \textit{pkt} \equiv \lambda (\text{d, dip, sip}). \textit{Pkt} \text{d} \text{dip} \text{sip}\

\textbf{lemma} \textit{pkt_simp} [simp]:
\textit{pkt}(\text{d, dip, sip}) = \textit{Pkt} \text{d} \text{dip} \text{sip}\
\text{unfolding} \textit{pkt_def} \text{by simp}\

\end{multicols}

\section{4.3 The AODV protocol}

\textbf{theory} \textit{D_Aodv}
\textbf{imports} \textit{D_Aodv_Data D_Aodv_Message}
\texttt{"../../../AWN/AWN_SOS_Labels" "../../../AWN/AWN_Invariants"}
\textbf{begin}\

\subsection{4.3.1 Data state}

\textbf{record} \textit{state} =
\begin{verbatim}
  \textit{ip} :: \"ip\"
  \textit{sn} :: \"sqn\"
  \textit{rt} :: \"rt\"
  \textit{rreqs} :: \"(ip \times rreqid) set\"
\end{verbatim}
abbreviation aodv_init :: "ip \Rightarrow 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),
p = (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),
handled= (SOME x. True)
|)"

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),
p := (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),
handled:= (SOME x. True)
|)"

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"
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' ∧
  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' ⇒
| _ ⇒ {}"

lemma is_rrep_asm [dest!]:
  assumes "ξ' ∈ is_rrep ξ"
  shows "∃ hops' dip' dsn' oip' sip'.
  msg ξ = Rrep hops' dip' dsn' oip' 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 =
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 pseqp =
  PAodv | PNewPkt | PPkt | PRreq | PRrep | PRerr

fun nat_of_seqp :: "pseqp ⇒ 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, pseq, pseq 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)"
[\xi, \xi \circ \text{rt} := \text{update (rt } \xi) (\text{sip } \xi) (0, \text{unk, val, 1, sip } \xi, \{\})] \\
\text{RREQ(\lambda_\xi. (hops } \xi, \text{rreqid } \xi, \text{dip } \xi, \text{dsn } \xi, \text{oip } \xi, \text{osn } \xi, \text{sip } \xi, \text{handled } \xi))

\text{\textbf{is\_rrep}}

\[
[\xi, \xi \circ \text{rt} := \text{update (rt } \xi) (\text{sip } \xi) (0, \text{unk, val, 1, sip } \xi, \{\})]
\]

\text{RREP(\lambda_\xi. (hops } \xi, \text{dip } \xi, \text{dsn } \xi, \text{oip } \xi, \text{sip } \xi))

\text{\textbf{is\_rerr}}

\[
[\xi, \xi \circ \text{rt} := \text{update (rt } \xi) (\text{sip } \xi) (0, \text{unk, val, 1, sip } \xi, \{\})]
\]

\text{RERR(\lambda_\xi. (dests } \xi, \text{sip } \xi))

\text{\textbf{AODV}}

\[
\langle \lambda_\xi, \{ \xi \circ \text{dests} := (\text{\lambda\text{rrip. if (rip } \in \text{vd (rt } \xi) \land \text{nhop (rt } \xi) \text{ rip } \neq \text{dests } \xi) \} \text{\ λ\text{rreq (0, rreqid } \xi, \text{ip } \xi, \text{sqn (rt } \xi) (\text{dip } \xi), \text{sqnf (rt } \xi) (\text{dip } \xi), \text{ip } \xi, \text{sn } \xi, \text{ip } \xi, \text{False}}). \text{AODV()}\rangle
\]

\text{\textbf{PNewPkt}}

\[
\langle \xi, \text{dip } \xi = \text{ip } \xi \rangle
\]

\text{deliver(\lambda_\xi. \text{data } \xi).AODV()}

\text{\textbf{PPkt}}

\[
\langle \xi, \text{dip } \xi \neq \text{ip } \xi \rangle
\]

\[
\langle \xi, \text{dip } \xi \in \text{vd (rt } \xi) \rangle
\]

\text{unicast(\lambda_\xi. \text{the (nhop (rt } \xi) (\text{dip } \xi)), \lambda_\xi. \text{pkt(data } \xi, \text{dip } \xi, \text{oip } \xi)).AODV()}

\text{\textbf{groupcast(\lambda_\xi. pre } \xi, \lambda_\xi. \text{rerr(dests } \xi, \text{ip } \xi)).AODV()}}

\text{\textbf{AODV}}

\[
\langle \xi, \text{dip } \xi \neq \text{ip } \xi \rangle
\]

\[
\langle \xi, \text{dip } \xi \neq \text{ip } \xi \rangle
\]

\[
\langle \xi, \text{dip } \xi \neq \text{ip } \xi \rangle
\]

353
Γ′_AODV_ PRreq = labelled PRreq (  
(ξ. (oip ξ, rreqid ξ) ∈ rreqs ξ)  
AODV()  
① (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()  
  ② (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 ⊢ 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()  
  ③ (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())  
  )  
  ④ (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()  
  )  
  )  
  ⑤ (ξ. (oip ξ, rreqid ξ) ∈ rreqs ξ)  
  [ξ. ξ ⊢ rt := update (rt ξ) (dip ξ) (dsn ξ, kno, val, hops ξ + 1, sip ξ, {})]  
  (  
  (ξ. oip ξ = ip ξ)  
  AODV()  
  ⑥ (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)})}]  
  )

Γ′_AODV_ 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()  
  ⑦ (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)})])

354
x. 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. 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. 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()
)"

"Γ_AODV PRerr = labelled PRerr {
 senators := (λrip. if (rip ∈ vD (rt ξ) ∧ nhop (rt ξ) rip = nhop (rt ξ) (oip ξ))
then Some (inc (sqn (rt ξ) rip)) else None) 
senators := (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)
store := setRRF (store ξ) (dests ξ)
pre := ⋃ { the (precs (rt ξ) rip) | rip. rip ∈ dom (dests 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()
"  AODV()

lemma Γ_AODV.simps [simp del, code del]
lemmas Γ_AODV.simps [simp, code] = Γ_AODV.simps [simplified]

fun Γ_AODV_skeleton where
  "Γ_AODV_skeleton PAodv = seqp_skeleton (Γ_AODV PAodv)"
/ "Γ_AODV_skeleton PNewPkt = seqp_skeleton (Γ_AODV PNewPkt)"
/ "Γ_AODV_skeleton PPkt = seqp_skeleton (Γ_AODV PPkt)"
/ "Γ_AODV_skeleton PRreq = seqp_skeleton (Γ_AODV PRreq)"
/ "Γ_AODV_skeleton PRrep = seqp_skeleton (Γ_AODV PRrep)"
/ "Γ_AODV_skeleton PRerr = seqp_skeleton (Γ_AODV PRerr)"

lemma Γ_AODV_skeleton_wf [simp]:
"wellformed Γ_AODV_skeleton"
proof (rule, intro allI)
  fix pn pn'
  show "call(pn') ∉ sterm sl (Γ_AODV_skeleton pn)"
    by (cases pn) simp_all
qed

declare Γ_AODV_skeleton.simps [simp del, code del]
lemmas Γ_AODV_skeleton.simps [simp, code]
  = Γ_AODV_skeleton.simps [simplified Γ_AODV.simps seqp_skeleton.simps]

lemma aodv_proc_cases [dest]:
  fixes p pn
  shows "p ∈ cterm sl (Γ_AODV pn) "
\[ p \in \text{cterms} (\Gamma_{AODV} \text{PRrep}) \lor p \in \text{cterms} (\Gamma_{AODV} \text{PRerr}) \]

using assms
by (cases pn) simp_all

definition \( \sigma \_AODV \) :: "ip \Rightarrow (\text{state} \times (\text{state}, \text{msg}, \text{pseqp}, \text{pseqp label}) \text{ seqp}) \text{ set}"
where "\( \sigma \_AODV \ i \equiv \{(\text{aodv}_\text{init} \ i, \ \Gamma_{AODV} \text{PAodv})\}\)"

abbreviation \( \text{paodv} \) :: "ip \Rightarrow (\text{state} \times (\text{state}, \text{msg}, \text{pseqp}, \text{pseqp label}) \text{ seqp}, \text{ msg seq_action}) \text{ automaton}"
where "\( \text{paodv} \ i \equiv (|\text{init} = \sigma \_AODV \ i, \text{trans} = \text{seqp}_\text{sos} \Gamma_{AODV}|)\)"

lemma \( \text{aodv}_\text{trans} \): "\( \text{trans} (\text{paodv} \ i) = \text{seqp}_\text{sos} \Gamma_{AODV} \)"
by simp

lemma \( \text{aodv}_\text{control}_\text{within} \) [simp]: "\( \text{control}_\text{within} \Gamma_{AODV} (\text{init} (\text{paodv} \ i))\)"
unfolding \( \sigma \_AODV \text{ def} \) by (rule \( \text{control}_\text{withinI} \)) (auto simp del: \( \Gamma \text{AODV \_simps} \))

lemma \( \text{aodv}_\text{wf} \) [simp]: "\( \text{wellformed} \Gamma_{AODV} \)"
proof (rule, intro allI)
  fix \( p n n' \)
  show "\( \text{call} (p n') \notin \text{cterms} (\Gamma_{AODV} \text{pn})\)"
    by (cases pn) simp_all
qed

lemmas \( \text{aodv}_\text{labels}_\text{not}_\text{empty} \) [simp] = \( \text{labels}_\text{not}_\text{empty} \ [\text{OF aodv}_\text{wf}] \)

lemma \( \text{aodv}_\text{ex}_\text{label} \) [intro]: "\( \exists l. l \in \text{labels} \Gamma_{AODV} p \)"
by (metis \( \text{aodv}_\text{labels}_\text{not}_\text{empty} \ \text{all}_\text{not}_\text{in}_\text{conv} \))

lemma \( \text{aodv}_\text{ex}_\text{labelE} \) [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 \( \text{aodv}_\text{ex}_\text{label} \))

lemma \( \text{aodv}_\text{simple}_\text{labels} \) [simp]: "\( \text{simple}_\text{labels} \Gamma_{AODV} \)"
proof
  fix \( p n \)
  assume "\( p \in \text{subterms}(\Gamma_{AODV} \text{pn})\)"
  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]: "\( (\xi, p) \in \sigma \_AODV \ i \implies \text{labels} \Gamma_{AODV} p = \{\text{PAodv}-:0\}\)"
unfolding \( \sigma \_AODV \text{ def} \) by simp

lemma \( \text{aodv}_\text{init}_\text{kd}_\text{empty} \) [simp]:
  "\( (\xi, p) \in \sigma \_AODV \ i \implies \text{kd} (\text{rt} \xi) = \{\}\)"
unfolding \( \sigma \_AODV \text{ def} \) \( \text{kd}_\text{def} \) by simp

lemma \( \text{aodv}_\text{init}_\text{sip}_\text{not}_\text{ip} \) [simp]: "\( \neg (\text{sip} (\text{aodv}_\text{init} \ i) = i) \)" by simp

lemma \( \text{aodv}_\text{init}_\text{sip}_\text{not}_\text{ip}' \) [simp]:
  assumes "\( (\xi, p) \in \sigma \_AODV \ i \)"
  shows "\( \text{sip} \xi \neq \text{ip} \xi \)"
using assms unfolding \( \sigma \_AODV \text{ def} \) by simp

lemma \( \text{aodv}_\text{init}_\text{sip}_\text{not}_\text{i} \) [simp]:
  assumes "\( (\xi, p) \in \sigma \_AODV \ i \)"
  shows "\( \text{sip} \xi \neq i \)"
using assms unfolding \( \sigma \_AODV \text{ def} \) by simp
lemma clear_locals_sip_not_ip':
  assumes "ip \( \xi \) = i"
  shows "\(~(\text{sip (clear_locals } \xi \text{) = i})\)"
  using assms by auto

Stop the simplifier from descending into process terms.

declare seq_congs [cong]

Configure the main invariant tactic for AODV.

declare
  \( \Gamma_{AODV} \_\text{simps} \) [cterms_env]
  \( \text{aodv_proc_cases} \) [ctermsl_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

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 \text{case m of } \text{Pkt } \_\_\_ \Rightarrow \text{False} \text{ | } _ \Rightarrow \text{True}"

definition msg_sender :: "msg \Rightarrow ip"
where "msg_sender m \equiv \text{case m of } \text{Rreq } \_\_\_\_\_\_\_\_\_\_\_ \Rightarrow \text{ip} \text{ | } \text{Rrep } \_\_\_\_ \Rightarrow \text{ip} \text{ | } \text{Rerr } \_ \Rightarrow \text{ip} \text{ | } \text{Pkt } \_\_ \Rightarrow \text{ip}"

lemma msg_sender_simps [simp]:
  "\( \forall \text{hops rreqid dip dsn dsk osn sip handled.} \)\
  \( \text{msg_sender (Rreq } \text{hops rreqid dip dsn dsk osn sip handled) = sip} \)"
  "\( \forall \text{hops dip dsn oip sip.} \)\
  \( \text{msg_sender (Rrep } \text{hops dip dsn oip sip) = sip} \)"
  "\( \forall \text{dests sip.} \)\
  \( \text{msg_sender (Rerr dests sip) = sip} \)"
  "\( \forall \text{d dip sip.} \)\
  \( \text{msg_sender (Pkt } \text{d dip sip) = sip} \)"

unfolding msg_sender_def by simp_all

definition msg_zhops :: "msg \Rightarrow bool"
where "msg_zhops m \equiv \text{case m of } \text{Rreq } \_\_\_\_\_\_\_\_\_\_\_ osnc _ _ \Rightarrow \text{osnc} \geq 1 \text{ | } \text{Rrep } \_\_\_\_ \Rightarrow \text{dip} \geq 1 \text{ | } _ \Rightarrow \text{True}"

lemma msg_zhops_simps [simp]:
  "\( \forall \text{hops rreqid dip dsn dsk oip osn sip handled.} \)\
  \( \text{msg_zhops (Rreq } \text{hops rreqid dip dsn dsk oip osn sip handled) = (hops = 0 \rightarrow oip = sip)} \)"
  "\( \forall \text{hops dip dsn oip sip.} \)\
  \( \text{msg_zhops (Rrep } \text{hops dip dsn oip sip) = (hops = 0 \rightarrow dip = sip)} \)"
  "\( \forall \text{dests sip.} \)\
  \( \text{msg_zhops (Rerr dests sip) = True} \)"
  "\( \forall \text{d dip.} \)\
  \( \text{msg_zhops (Newpkt } \text{d dip) = True} \)"
  "\( \forall \text{d dip sip.} \)\
  \( \text{msg_zhops (Pkt } \text{d dip sip) = True} \)"

unfolding msg_zhops_def by simp_all

definition rreq_rrep_sn :: "msg \Rightarrow bool"
where "rreq_rrep_sn m \equiv \text{case m of } \text{Rreq } \_\_\_\_\_\_\_\_\_\_\_ osnc _ _ \Rightarrow \text{osnc} \geq 1 \text{ | } \text{Rrep } \_\_\_\_ \Rightarrow \text{dsk} \geq 1 \text{ | } _ \Rightarrow \text{True}"

end
lemma rreq_rrep_sn_simps [simp]:

\[ \forall \text{hops rreqid dip dsn osn sip handled.} \]
\[ \text{rreq_rrep_sn (Rreq hops rreqid dip dsn osn sip handled) = (osn \geq 1)} \]
\[ \forall \text{hops dip dsn osn sip.} \quad \text{rreq_rrep_sn (Rrep hops dip dsn osn sip) = (dsn \geq 1)} \]
\[ \forall \text{dests sip.} \quad \text{rreq_rrep_sn (Rerr dests sip) = True} \]
\[ \forall \text{d dip.} \quad \text{rreq_rrep_sn (Newpkt d dip) = True} \]
\[ \forall \text{d dip sip.} \quad \text{rreq_rrep_sn (Pkt d dip sip) = True} \]

unfolding rreq_rrep_sn_def by simp_all

definition rreq_rrep_fresh :: "rt \Rightarrow msg \Rightarrow bool"
where "rreq_rrep_fresh crt m ≡ case m of Rreq hopsc _ _ _ _ oipc osnc ipcc _ ⇒ (ipcc \neq oipc \rightarrow oipc \in kD(crt) \land (sqn crt oipc > osnc \\
\lor (sqn crt oipc = osnc \land the (dhops crt oipc) \leq hopsc \\
\lor the (flag crt oipc) = val)))
| Rrep hopsc dipc dsnc _ ipcc ⇒ (ipcc \neq dipc \rightarrow dipc \in kD(crt) \\
\land sqn crt dipc = dsnc \\
\land the (dhops crt dipc) = hopsc \\
\land the (flag crt dipc) = val)
| _ ⇒ True"

lemma rreq_rrep_fresh [simp]:

\[ \forall \text{hops rreqid dip dsn osn sip handled.} \]
\[ \text{rreq_rrep_fresh crt (Rreq hops rreqid dip dsn osn sip handled) =} \]
\[ (\text{sip \neq oip} \rightarrow \text{oip} \in kD(crt)) \\
\land (\text{sqn crt oip} > \text{osn} \\
\lor (\text{sqn crt oip} = \text{osn} \\
\land \text{the (dhops crt oip)} \leq \text{hops} \\
\lor \text{the (flag crt oip) = val})\))
\]
\[ \forall \text{hops dip dsn osn sip.} \quad \text{rreq_rrep_fresh crt (Rrep hops dip dsn osn sip) =} \]
\[ (\text{sip \neq dip} \rightarrow \text{dip} \in kD(crt)) \\
\land \text{sqn crt dip = dsn} \\
\land \text{the (dhops crt dip) = hops} \\
\land \text{the (flag crt dip) = val})\)
\[ \forall \text{dests sip.} \quad \text{rreq_rrep_fresh crt (Rerr dests sip) = True} \]
\[ \forall \text{d dip.} \quad \text{rreq_rrep_fresh crt (Newpkt d dip) = True} \]
\[ \forall \text{d dip sip.} \quad \text{rreq_rrep_fresh crt (Pkt d dip sip) = True} \]

unfolding rreq_rrep_fresh_def by simp_all

definition rerr_invalid :: "rt \Rightarrow msg \Rightarrow bool"
where "rerr_invalid crt m ≡ case m of Rerr destsc _ ⇒ (\forall ripc \in \text{dom(destsc)}. \text{ripc} \in iD(crt) \land \text{the (destsc ripc) = sqn crt ripc})
| _ ⇒ True"

lemma rerr_invalid [simp]:

\[ \forall \text{hops rreqid dip dsn osn sip handled.} \]
\[ \text{rerr_invalid crt (Rreq hops rreqid dip dsn osn sip handled) = True} \]
\[ \forall \text{hops dip dsn osn sip.} \quad \text{rerr_invalid crt (Rrep hops dip dsn osn sip) = True} \]
\[ \forall \text{dests sip.} \quad \text{rerr_invalid crt (Rerr dests sip) = (\forall ripc \in \text{dom(dests)}. \text{ripc} \in iD(crt) \land \text{the (dests ripc) = sqn crt ripc})} \]
\[ \forall \text{d dip.} \quad \text{rerr_invalid crt (Newpkt d dip) = True} \]
\[ \forall \text{d dip sip.} \quad \text{rerr_invalid crt (Pkt d dip sip) = True} \]

unfolding rerr_invalid_def by simp_all

definition initmissing :: "(\text{nat} \Rightarrow \text{state option}) \times \text{a} \Rightarrow (\text{nat} \Rightarrow \text{state}) \times \text{a}"
where "initmissing σ = (\lambda i. \text{case (fст σ) i of None \Rightarrow aodv_init i \mid Some s \Rightarrow s, snd σ})"

lemma not_in_net_ips_fst_init_missing [simp]:

assumes "i /\in \text{net_ips} \sigma"

shows "\text{fst (initmissing (netgmap fst σ)) i = aodv_init i}"

358
lemma \texttt{fst\_initmissing\_netgmap\_pair\_fst \ [simp]}:
\texttt{"fst (initmissing (netgmap (\lambda(p, q). (fst (id p), snd (id p), q)) s)) = fst (initmissing (netmap fst s))"}
unfolding initmissing\_def by auto

We introduce a streamlined alternative to \texttt{initmissing} with \texttt{netgmap} to simplify invariant statements and thus facilitate their comprehension and presentation.

lemma \texttt{fst\_initmissing\_netgmap\_default\_aodv\_init\_netlift}:
\texttt{"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
\texttt{netglobal :: \((\text{nat} \Rightarrow \text{state}) \Rightarrow \text{bool}\) \Rightarrow ((\text{state} \times 'b) \times 'c) net\_state \Rightarrow bool"}
where
\texttt{"netglobal P \equiv (\lambda s. P (default aodv\_init (netlift fst s)))"}
end

4.5 Quality relations between routes

theory D\_Fresher
imports D\_Aodv\_Data
begin

4.5.1 Net sequence numbers

On individual routes
definition
\texttt{nsqn\_r :: \"r \Rightarrow sqn"}
where
\texttt{"nsqn\_r, r \equiv if \pi_4(r) = val \lor \pi_2(r) = 0 then \pi_2(r) else (\pi_2(r) - 1)"}

lemma \texttt{nsqn\_r\_def'}:
\texttt{"nsqn\_r, r = (if \pi_4(r) = inv \then \pi_2(r) - 1 \else \pi_2(r))"}
unfolding nsqn\_r\_def by simp

lemma \texttt{nsqn\_r\_zero \ [simp]}:
\texttt{"\\&\& dsn dsk flag hops nhip pre. nsqn\_r (0, dsk, flag, hops, nhip, pre) = 0"}
unfolding nsqn\_r\_def by clarsimp

lemma \texttt{nsqn\_r\_val \ [simp]}:
\texttt{"\\&\& dsn dsk hops nhip pre. nsqn\_r (dsn, dsk, val, hops, nhip, pre) = dsn"}
unfolding nsqn\_r\_def by clarsimp

lemma \texttt{nsqn\_r\_inv \ [simp]}:
\texttt{"\\&\& dsn dsk hops nhip pre. nsqn\_r (dsn, dsk, inv, hops, nhip, pre) = dsn - 1"}
unfolding nsqn\_r\_def by clarsimp

lemma \texttt{nsqn\_r\_lte\_dsn \ [simp]}:
\texttt{"\\&\& 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
\texttt{nsqn :: \"rt \Rightarrow ip \Rightarrow sqn"}
where
\texttt{"nsqn \equiv \lambda rt dip. case \sigma\_route(rt, dip) of None \Rightarrow 0 | Some r \Rightarrow nsqn\_r(r)"}

lemma \texttt{nsqn\_sqn\_def}:
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 nsqnr_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]:
  \(\forall rt dip \in kD(rt) = \Rightarrow nsqn r (\text{the (\text{addpreRT} rt dip npre dip'))} = nsqn r (\text{the (rt dip'})}\)
  unfolding addpreRT_def nsqn_r_def
  by (frule kD_Some) (clarsimp split: option.split)

lemma sqn_nsqn:
  \(\forall rt dip. \text{sqn rt dip} - 1 \leq \text{nsqn rt dip}\)
  unfolding sqn_def nsqn_def
  by (clarsimp split: option.split)

lemma sqn_nsqn:
  "sqn rt dip \leq nsqn rt dip"
  unfolding sqn_def nsqn_def by (clarsimp split: option.split)

lemma nsqnr_idx_nsqn [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 - 1"
  proof -
  from "ip \in vD(rt)" have "ip \in kD(rt)"
  and "the (flag rt ip) = val" by auto
  thus \(\text{thesis}\) ..
  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 \(\text{thesis}\) ..
  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)

lemma nsqn_addpreRT_inv [simp]:
"∀ rt dip npre dip'. dip ∈ kD(rt) ⇒
sqn (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
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)

4.5.2 Comparing routes

definition fresher :: "r ⇒ r ⇒ bool" ("(_ ⊑ _)" [51, 51] 50)
where
"fresher r r' ≡ ((nsqn, r < nsqn, r) ∨ (nsqn, r = nsqn, r', π₅(r) ≥ π₅(r')))"

lemma fresherI1 [intro]:
assumes "nsqn, r < nsqn, r'"
shows "r ⊑ r'"
unfolding fresher_def using assms by simp

lemma fresherI2 [intro]:
assumes "nsqn, r = nsqn, r'" and "π₅(r) ≥ π₅(r')"
shows "r ⊑ r'"
unfolding fresher_def using assms by simp

lemma fresherI [intro]:
assumes "(nsqn, r < nsqn, r') ∨ (nsqn, r = nsqn, r', π₅(r) ≥ π₅(r'))"
shows "r ⊑ r'"
unfolding fresher_def using assms.
lemma fresherE [elim]:
assumes "r ⊑ r'"
and "nsqn r < nsqn r' ⩾ P r r'"
and "nsqn r = nsqn r' ∧ π₅(r) ⩾ π₅(r') ⩾ P r r'"
says "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'
says "(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

4.5.3 Comparing routing tables

definition
rt_fresher :: "ip ⇒ rt ⇒ rt ⇒ bool"
where
"rt_fresher ≡ λdip rt rt'. (the (σ_route(rt, dip))) ⊑ (the (σ_route(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_fresherI1 [intro]:
assumes "dip ∈ kD(rt1)"
and "dip ∈ kD(rt2)"
362

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 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) 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) ⊑ the (rt2 dip)"
and "the (rt2 dip) ⊆ 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 "π₅(the (rt dip)) = π₅(the (rt' dip))"
shows "rt ≈_{dip} rt'"
proof
from assms(1-2,4) have dhops': "the (dhops rt' dip) ≤ the (dhops rt dip)"
  by (simp add: proj5_eq_dhops)
with assms(1-3) show "rt ⊑_{dip} rt'"
  by (rule rt_fresherI2)
next
from assms(1-2,4) have dhops: "the (dhops rt dip) ≤ the (dhops rt' dip)"
  by (simp add: proj5_eq_dhops)
with assms(2,1) assms(3) [symmetric]
  show "rt' ⊑_{dip} rt"
  by (rule rt_fresherI2)
qed

lemma rt_fresh_asE [elim]:
assumes "rt1 ≈_{dip} rt2"
  and "[ rt1 ⊑_{dip} rt2; rt2 ⊑_{dip} rt1 ] ⇒ 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 ≈_{dip} rt2"
shows "rt1 ⊑_{dip} rt2"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
assumes "rt1 ≈_{dip} rt2"
shows "rt2 ⊑_{dip} rt1"
using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
assumes "rt1 ≈_{dip} rt2"
shows "rt2 ≈_{dip} rt1"
using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
assumes "¬ (rt1 ⊑_{dip} rt2)"
shows "¬ (rt1 ≈_{dip} rt2)"
proof
  assume "rt1 ≈_{dip} rt2"
  hence "rt1 ⊑_{dip} rt2" ..
  with (¬ (rt1 ⊑_{dip} rt2)) show False ..
qed

lemma not_rt_fresh_asI2 [intro]:
assumes "¬ (rt2 ⊑_{dip} rt1)"
shows "¬ (rt1 ≈_{dip} rt2)"
proof
  assume "rt1 ≈_{dip} rt2"
  hence "rt2 ⊑_{dip} rt1" ..
  with (¬ (rt2 ⊑_{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_nsqr:
  assumes "dip ∈ kD(rt1)"
  and "dip ∈ kD(rt2)"
  and "rt1 ≈ dip rt2"
  shows "nsqn (the (rt2 dip)) = nsqn (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 dsn_n dsk_n f_n hops_n nhip_n pre_n
    where rt_n [simp]: "the (rt dip) = (dsn_n, dsk_n, f_n, hops_n, nhip_n, pre_n)"
    by (metis prod_cases6)
  with ⟨the (dhops rt dip) ≥ 1⟩ and ⟨dip∈kD(rt)⟩ have "hops_n ≥ 1"
by (metis proj5_eq_dhops proj4(4))
from ⟨dip ∈ kD(rt)⟩ assoc
have simp: "sqn rt dip = dsn n"
and simp: "the (dhops rt dip) = hops n"
and simp: "the (flag rt dip) = f n"
by (simp add: sqn_def proj5_eq_dhops [symmetric]
proj4_eq_flag [symmetric])

from ⟨update_arg_wf r⟩ assoc
have "(dsn n, dsk n, f n, hops n, nhip n, pre n) ⊑ the ((update rt dip r) dip)"
proof (rule wf_r_cases)
fix nhip pre
from ⟨hops n ≥ 1⟩ assoc
have "⋀ pre'. (dsn n, dsk n, f n, hops n, nhip n, pre n) ⊑ (dsn n, unk, val, Suc 0, nhip, pre')"
unfolding freshness_def sqn_def
by (cases f n) auto
thus "(dsn n, dsk n, f n, hops n, nhip n, pre n) ⊑ (dsn, kno, val, hops, nhip, pre ∪ pre n)"
using ⟨dip ∈ kD(rt)⟩ assoc
by (rule update_cases_kD, simp_all add: ⟨0 < dsn n⟩)

next
fix dsn :: sqn and hops nhip pre
assume "0 < dsn"
show "(dsn n, dsk n, f n, hops n, nhip n, pre n) ⊑ the (update rt dip (dsn, kno, val, hops, nhip, pre) dip)"
proof (rule update_cases_kD [OF ⟨dip ∈ kD(rt)⟩ assoc], simp_all add: ⟨0 < dsn n⟩)
assume "dsn n < dsn"
thus "(dsn n, dsk n, f n, hops n, nhip n, pre n) ⊑ (dsn, kno, val, hops, nhip, pre ∪ pre n)"
unfolding freshness_def by auto

next
assume "dsn n = dsn"
and "hops < hops n"
thus "(dsn n, dsk n, f n, hops n, nhip n, pre n) ⊑ (dsn, kno, val, hops, nhip, pre ∪ pre n)"
unfolding freshness_def nsqn_r_def by simp

next
assume "dsn n = dsn"
with ⟨0 < dsn n⟩ assoc
show "(dsn n, dsk n, inv, hops n, nhip n, pre n) ⊑ (dsn, kno, val, hops, nhip, pre ∪ pre n)"
unfolding freshness_def by simp
qed

hence "rt ⊑ dip update rt dip r"
by (rule single_rt_fresher, simp)
with ⟨dip = ip⟩ assoc show thesis assoc by simp

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)⟩ assoc
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 assoc
unfolding invalidate_def sqn_def
by (rule single_rt_fresher, auto simp: freshness_def)

qed
lemma \textit{nsqn}\textsubscript{r} invalidate [simp]:
assumes "\(dip \in kD(rt)\)"
and "\(dip \in \text{dom}(\text{dests})\)"
shows "\(\text{nsqn}_r (\text{the (invalidate } rt \text{ dests } dip)) = \text{the (dests } dip) - 1\)"
using \textit{assms}\ unfolding \textit{invalidate}\_def by auto

lemma \textit{rt fresh as inc invalidate} [simp]:
assumes "\(dip \in kD(rt)\)"
and "\(\forall ri dip \in \text{dom}(\text{dests}). ri \in vD(rt) \land \text{the (dests } ri) = \text{inc (sqn } rt ri)\)"
shows "\(rt \approx dip \text{ invalidate } rt \text{ dests}\)"
proof (cases "\(dip \in kD(\text{dests})\)"
assume "\(dip \not\in \text{dom}(\text{dests})\)"
with \(\langle \langle \ \rangle \\rangle \)
have "\(dip \in kD(\text{invalidate } rt \text{ dests})\)"
       by simp
with \(\langle \langle \ \rangle \\rangle \)
       show ?thesis
       by rule (simp\_all add: \dip\_dom\_dests)
next
assume "\(dip \in \text{dom}(\text{dests})\)"
with \textit{assms}(2) have "\(dip \in vD(rt)\)" by auto
from \(\langle \langle \ \rangle \\rangle \)
moreover then have "\(dip \in kD(\text{invalidate } rt \text{ dests})\)" by simp
ultimately show ?thesis
proof
from \(\langle \langle \ \rangle \\rangle \)
have "\(\text{nsqn } rt dip = \text{sqn } rt dip\)" by simp
also have "\(\text{sqn } rt dip = \text{nsqn}_r (\text{the (invalidate } rt \text{ dests } dip))\)"
proof -
from \(\langle \langle \ \rangle \\rangle \)
have "\(\text{nsqn}_r (\text{the (invalidate } rt \text{ dests } dip)) = \text{the (dests } dip) - 1\)"
       using \(\langle \langle \ \rangle \\rangle \)
       with \(\langle \langle \ \rangle \\rangle \)
       show "\(\text{sqn } rt dip = \text{nsqn}_r (\text{the (invalidate } rt \text{ dests } dip))\)" by simp
qed
also from \(\langle \langle \ \rangle \\rangle \)
have "\(\text{nsqn}_r (\text{the (invalidate } rt \text{ dests } dip)) = \text{nsqn (invalidate } rt \text{ dests) } dip\)"
       by (simp add: \kd\_nsqn)
finally show "\(\text{nsqn } rt dip = \text{nsqn (invalidate } rt \text{ dests) } dip\)".
qed simp

lemmas \textit{rt fresher inc invalidate} [simp] = \textit{rt fresh as inc invalidate} [THEN \textit{rt fresh asD1}]

lemma \textit{rt fresh as addpreRT} [simp]:
assumes "\(ip \in kD(rt)\)"
shows "\(rt \approx dip (\text{addpreRT } rt ip npre)\)"
using \textit{assms} [THEN \kd\_Some] by (auto simp: \textit{addpreRT}\_def)

lemmas \textit{rt fresher addpreRT} [simp] = \textit{rt fresh as addpreRT} [THEN \textit{rt fresh asD1}]

\subsection{4.5.4 Strictly comparing routing tables}
definition \textit{rt strictly fresher} :: "ip \Rightarrow rt \Rightarrow rt \Rightarrow bool"
where
"\text{rt strictly fresher} \equiv \lambda dip rt1 rt2. \text{(rt1 } \sqsubseteq dip \text{ rt2) } \land \neg (\text{rt1 } \approx dip \text{ rt2})"

abbreviation
\textit{rt strictly fresher} syn :: "rt \Rightarrow ip \Rightarrow rt \Rightarrow bool" ("\_/ _/ _") [51, 999, 51] 50
where
"rt1 \sqsubseteq dip rt2 \Rightarrow rt_\text{strictly fresher} i rt1 rt2"

lemma \textit{rt strictly fresher} def':
"\text{rt1 } \sqsubseteq dip rt2 = ((\text{rt1 } \sqsubseteq dip rt2) \land \neg (\text{rt2 } \sqsubseteq dip rt1))"
unfolding \textit{rt strictly fresher} def \textit{rt fresh as def} by auto

lemma \textit{rt strictly fresher}I' [intro]:
\newpage
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 =
(nsqn_r (the (rt1 i)) < nsqn_r (the (rt2 i))
 ∨ (nsqn_r (the (rt1 i)) = nsqn_r (the (rt2 i)) ∧ π_5(the (rt1 i)) > π_5(the (rt2 i))))"
unfolding rt_strictly_fresher_def" rt_fresher_def fresher_def by auto

lemma rt_strictly_fresher_fresherD [dest]:
assumes "rt1 ⊏_i dip rt2"
saves "the (rt1 dip) ⊑ 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 ⊏_i dip rt2"
saves "¬ rt1 ≈_i dip rt2"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
assumes "rt1 ⊏_i dip rt2"
and "rt2 ⊏_i dip rt3"
shows "rt1 ⊏_i dip rt3"
using assms proof -
from rt1 ⊏_i dip rt2: obtain "the (rt1 dip) ⊑ the (rt2 dip)" by auto
also from rt2 ⊏_i dip rt3: obtain "the (rt2 dip) ⊑ the (rt3 dip)" by auto
finally have "the (rt1 dip) ⊑ the (rt3 dip)" .
moreover have "¬ (rt1 ≈_i dip rt3)"
proof -
from rt1 ⊏_i dip rt2: obtain "¬(the (rt2 dip) ⊑ the (rt1 dip))" by auto
also from rt2 ⊏_i dip rt3: obtain "¬(the (rt3 dip) ⊑ the (rt2 dip))" by auto
finally have "¬(the (rt3 dip) ⊑ the (rt1 dip))" .
thus ?thesis ..
qed
ultimately show "rt1 ⊏_i dip rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: "¬ (rt ⊏_i dip rt)"
using assms unfolding rt_strictly_fresher_def
by clarsimp
lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
assumes "rt1 ⊏ dip rt2"
and "rt2 ⊑ dip rt3"
shows "rt1 ⊏ 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 \( \text{nsqn } rt1 \text{ dip} < \text{nsqn } rt2 \text{ dip} \) show "False"
by simp
qed
qed

lemma rt_strictly_fresher_eqI [intro]:
assumes "i \in kD(rt1)"
and "i \in kD(rt2)"
and "\text{nsqn } rt1 \text{ i} = \text{nsqn } rt2 \text{ i}"
and "\pi_5(\text{the } (rt2 \text{ i})) < \pi_5(\text{the } (rt1 \text{ i}))"
shows "rt1 \sqsubset rt2"
using assms unfolding rt_strictly_fresher_def' by (auto simp add: kD_nsqn)

lemma invalidate_rtsf_left [simp]:
"\wedge \text{dests dip rt'}. \text{dests dip} = \text{None } \Longrightarrow \text{invalidate rt \text{ dests} } \sqsubset \text{dip rt'}" = (rt \sqsubset \text{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(\text{invalidate rt1 \text{ dests})}"
shows "(\text{invalidate rt1 \text{ dests} } \sqsubset \text{dip rt2}) = (\text{rt1 } \sqsubset \text{dip rt2})" using assms unfolding invalidate_def vD_def
by clarsimp (metis (hide_lams, no_types) assms vD_Some vD_invalidate_vD_not_dests)

lemma rt_strictly_fresher_update_other [elim!]:
"\wedge \text{ip rt rt'}. \text{ip } rt \neq \text{ip rt'} \implies \text{update rt \text{ ip} r } \sqsubset \text{dip rt}'"
unfolding rt_strictly_fresher_def'
proof (clarsimp)
lemma addpreRT_strictly_fresher [simp]:
assumes "dip \in kD(rt)"
shows "\text{(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 "*": "\text{osn } < \text{sqn (rt2 nhip) dip}"
and "**": "rt \neq \text{update rt dip (osn, kno, val, hops, nhip, {})}" shows "\text{update rt dip (osn, kno, val, hops, nhip, {}) } \sqsubset \text{dip rt2 nhip}"
unfolding rt_strictly_fresher_def'
proof (rule disjI1)
from ** have "\text{nsqn (update rt dip (osn, kno, val, hops, nhip, {}) dip) } = \text{osn}"
by (rule nsqn_update_changed_kno_val)
with dip\in vD (rt2 nhip)
have "\text{nsqn, (the (update rt dip (osn, kno, val, hops, nhip, {}) dip)) } = \text{osn}"
by (simp add: kD_nsqn)
also have "\text{osn } < \text{sqn (rt2 nhip) dip}" by (rule *)
also have "\text{sqn (rt2 nhip) dip } = \text{nsqn, (the (rt2 nhip dip))}"
unfolding nsqn, def using dip \in vD (rt2 nhip)
by - (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
finally show "\text{nsqn, (the (update rt dip (osn, kno, val, hops, nhip, {}) dip)) } < \text{nsqn, (the (rt2 nhip dip))}".

370
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 "nsqn (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 = nsqn (the (rt2 nhip dip))"
    unfolding nsqn_def using ⟨dip ∈ vD(rt2 nhip)⟩
    by (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show "nsqn (the (update rt dip (osn, kno, val, Suc hops, nhip, {}) dip)) = nsqn (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

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 ||= onll Γ_AODV (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
  by inv_cterms

lemma sequence_number_one_or_bigger:
  "paodv i ||= onll Γ_AODV (λ(ξ, _). 1 ≤ sn ξ)"
  by (rule onll_step_to_invariantI [OF sequence_number_increases])
We can get rid of the onl/onll if desired...

lemma sequence_number_increases':
"paodv i |=_A (λ((ξ, _), _), (ξ', _) :: state. 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 step_invariant_weakenE [OF sequence_number_one_or_bigger]) auto

lemma sip_in_kD:
"paodv i ||=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 ||=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 ||=λ(ξ, l). (l ∈ {PRreq-:18..PRreq-:20} ∪ {PRreq-:2..PRreq-:6} −→ dip ξ ∈ kD (rt ξ)) ∧ (l ∈ {PRreq-:3..PRreq-:19} −→ oip ξ ∈ kD (rt ξ))"
by inv_cterms

Proposition 7.38
lemma includes_nhip:
"paodv i ||=λ(ξ, l). ∀ dip ∈ kD (rt ξ). the (nhop (rt ξ) dip) ∈ kD (rt ξ)
∧ (l = PRrep-:5 −→ (the (nhop (rt ξ) dip)) 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 (sn, kno, val, Suc 0, ip, {})})"
  hence "∀ dip ∈ kD (rt ξ).
    the (nhop (update (rt ξ) ip (sn, kno, val, Suc 0, ip, {})) dip) = ip
    ∨ the (nhop (update (rt ξ) ip (sn, kno, 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, {})) dip) = ip
  ∨ the (nhop (update (rt ξ) ip (sn, kno, val, Suc hops, sip, {})) dip) ∈ 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 aodv_wf sip_in_kD]
      onl_invariant_sterms [OF aodv_wf addpreRT_partly_welldefined]
      solve: one_hop nhip_is_sip)
qed

Proposition 7.22: needed in Proposition 7.4
lemma addpreRT_welldefined:
"paodv i ||=λ(ξ, l). (l ∈ {PRreq-:18..PRreq-:20} ∪ {PRreq-:2..PRreq-:6} −→ dip ξ ∈ kD (rt ξ))
∧ (l = PRreq-:19 −→ oip ξ ∈ kD (rt ξ))
∧ (l = PRrep-:5 −→ dip ξ ∈ kD (rt ξ))
∧ (l = PRrep-:6 −→ (the (nhop (rt ξ) (dip ξ)) dip) ∈ kD (rt ξ)))"
(is "_|= onl Γ_AODV ?P")
unfolding invariant_def

372
proof
fix s
assumes "s ∈ reachable (paodv i) TT"
then obtain ξ p where "s = (ξ, p)"
and "(ξ, p) ∈ reachable (paodv i) TT"
by (metis PairE)
have "onl Γ_AODV ?P (ξ, p)"
proof (rule onlI)

fix l
assumes "l ∈ labels Γ_AODV p"
with ⟨⟨ξ, p⟩ ∈ reachable (paodv i) TT⟩ ⟦l ∈ labels Γ_AODV 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 Γ_AODV ?P s"
by simp
qed

Proposition 7.4
lemma known_destinations_increase:
"paodv i ||= A onll Γ_AODV (λ((ξ, _), _, (ξ', _)). kD (rt ξ) ⊆ kD (rt ξ'))"
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 ||= A onll Γ_AODV (λ((ξ, _), _, (ξ', _)). rreqs ξ ⊆ rreqs ξ')"
by (inv_cterms simp add: subset_insertI)

lemma dests_bigger_than_sqn:
"paodv i ||= onl Γ_AODV (λ(ξ, 1). l ∈ {PAodv-:15..PAodv-: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:
"paodv i ||= A onll Γ_AODV (λ((ξ, _), _, (ξ', _)). ∀ ip. sqn (rt ξ) ip ≤ sqn (rt ξ') ip)"
proof -
{ fix \(\xi\) :: state
  assume *: "\(\forall ip \in \text{dom(dests } \xi\). ip \in kD (rt \xi) \land sqn (rt \xi) ip \leq \text{ the (dests } \xi \text{ ip})"  
  have "\(\forall ip. \text{ sqn (rt } \xi\) ip \leq \text{ sqn (invalidate (rt } \xi\) (dests } \xi\) ip)"
proof
  fix ip
  from * have "ip \notin \text{dom(dests } \xi\) \lor sqn (rt \xi) ip \leq \text{ the (dests } \xi \text{ ip})" by simp 
  thus "sqn (rt \xi) ip \leq \text{ sqn (invalidate (rt } \xi\) (dests } \xi\) ip)"
  by (metis \text{domI invalidate_sqn option.sel) 
qed } note solve_invalidate = this 
qed


Proposition 7.7

\text{lemma ip_constant:}
"paodv i \models A \Gamma_{AODV} (\lambda (\xi, \_). ip \xi = i)"
by (inv_cterms simp add: \sigma_{AODV}_def) 

Proposition 7.8

\text{lemma sender_ip_valid':}
"paodv i \models A \Gamma_{AODV} (\lambda ((\xi, \_), a, \_). \text{anycast (\lambda m. not_Pkt m \rightarrow msg_sender m = ip } \xi \text{ a)})"
by inv_cterms 

\text{lemma sender_ip_valid:}
"paodv i \models A \Gamma_{AODV} (\lambda ((\xi, \_), a, \_). \text{anycast (\lambda m. not_Pkt m \rightarrow msg_sender m = i) a})"
by (rule step_invariant_weaken_with_invariantE [OF ip_constant sender_ip_valid']) 
(auto dest!: onlD onllD) 

\text{lemma received_msg_inv:}
"paodv i \models (recvmsg P \rightarrow) \Gamma_{AODV} (\lambda (\xi, l). l \in \{PAodv-:1\} \rightarrow P (msg } \xi\)"
by inv_cterms 

Proposition 7.9

\text{lemma sip_not_ip':}
"paodv i \models (recvmsg (\lambda m. not_Pkt m \rightarrow msg_sender m \neq i) \rightarrow) \Gamma_{AODV} (\lambda (\xi, \_). \text{sip } \xi \neq ip \xi)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv] onl_invariant_sterms [OF aodv_wf ip_constant [THEN invariant_restrict_inD]] simp add: clear_locals_sip_not_ip')clarsimp+ 

\text{lemma sip_not_ip:}
"paodv i \models (recvmsg (\lambda m. not_Pkt m \rightarrow msg_sender m \neq i) \rightarrow) \Gamma_{AODV} (\lambda (\xi, \_). \text{sip } \xi \neq i)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv] onl_invariant_sterms [OF aodv_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

\text{lemma hop_count_positive:}
"paodv i \models \Gamma_{AODV} (\lambda (\xi, \_). \forall ip \in kD (rt } \xi\) \land (dhops (rt } \xi\) ip) \geq 1)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined]) auto 

\text{lemma reqq_dip_in_vD_dip_eq_ip:}
"paodv i \models \Gamma_{AODV} (\lambda (\xi, l). (l \in \{\text{PRreq-:18..PRreq-:21}\} \rightarrow dip } \xi \in vD(\xi)) \land (l \in \{\text{PRreq-:6, PRreq-:7}\} \rightarrow dip } \xi = ip \xi) \land (l \in \{\text{PRreq-:17..PRreq-:21}\} \rightarrow dip } \xi \neq ip \xi))"
proof (inv_cterms, elim conjE) 
fix } \xi pp p'
assume "((\xi, pp) \in \text{reachable (paodv } i) TT)"
and "\{PRreq-:19\}[\lambda \xi. \xi(\rt := \text{(addpreRT}(\rt \xi)(\oip \xi)\{\text{(nhop}(\rt \xi)(\dip \xi)\))))]} p' \in \text{sterms } \Gamma_{\text{AODV}} pp"
and "1 = PRreq-:19"
and "\dip \xi \in \text{vD} (\rt \xi)"
from this (1-3) have "\oip \xi \in kD (\rt \xi)"
by (auto dest: onl_invariant_sterms [OF aodv_wf addpreRT_welldefined, where l="PRreq-:19"])
with \(\dip \xi \in \text{vD} (\rt \xi)\);
show "\dip \xi \in \text{vD} (\text{(addpreRT}(\rt \xi)(\oip \xi)\{\text{(nhop}(\rt \xi)(\dip \xi)\))))" by simp
qed

Proposition 7.11
lemma anycast_msg_zhops:
"\rreqid \dip \dsn \dsk \oip \osn \sip.
\paodv i \mid_A onl \\Gamma_{\text{AODV}} (\\lambda(\_, \_). \text{anycast msg_zhops a})"
proof (inv_cterms inv add:
onl_invariant_sterms [OF aodv_wf received_msg_inv]
onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]],elim conjE)
fix l \xi a pp p' pp'
assume 
Pp \in \text{reachable } (\paodv i) TT
and "\{PRreq-:20\}unicast(\lambda \xi. \text{the}(\text{nhop}(\rt \xi)(\oip \xi)),
\lambda \xi. \text{Rrep}(\text{(dhops}(\rt \xi)(\dip \xi))(\dip \xi))(\text{sqn}(\rt \xi)(\dip \xi))(\oip \xi) (\ip \xi)).
p' \triangleright pp' \in \text{sterms } \Gamma_{\text{AODV}} pp"
and "1 = PRreq-:20"
and "a = unicast (\text{the}(\text{nhop}(\rt \xi)(\oip \xi)))
\text{(Rrep}(\text{(dhops}(\rt \xi)(\dip \xi))(\dip \xi))(\text{sqn}(\rt \xi)(\dip \xi))(\oip \xi) (\ip \xi))"
and *: 
\forall \ip \in kD (\rt \xi). \text{Suc} \text{0} \leq \text{the}(\text{dhops}(\rt \xi)\ip)"
and "\dip \xi \in \text{vD} (\rt \xi)"
from \(\dip \xi \in \text{vD} (\rt \xi)\) have "\dip \xi \in kD (\rt \xi)"
by (rule vD_iD_gives_kD(1))
with * have "\text{Suc} \text{0} \leq \text{the}(\text{dhops}(\rt \xi)(\dip \xi))" ..
thus "0 < \text{the}(\text{dhops}(\rt \xi)(\dip \xi))" by simp
qed

lemma hop_count_zero_oip_dip_sip:
"\paodv i \mid (\text{recvmsg } \text{msg_zhops } \rightarrow) onl \\Gamma_{\text{AODV}} (\lambda(\xi, 1).
(1 \in \{\text{PAodv-:4,PAodv-:5} \cup \{\text{PRreq-:n}|n. \text{True}\} \rightarrow
(\text{hops } \xi = \text{0} \rightarrow \oip \xi = \text{sip } \xi))
\land
((1 \in \{\text{PAodv-:6,PAodv-:7} \cup \{\text{PRrep-:n}|n. \text{True}\} \rightarrow
(\text{hops } \xi = \text{0} \rightarrow \dip \xi = \text{sip } \xi))))"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) auto

lemma osn_rreq:
"\paodv i \mid (\text{recvmsg } \text{rreq_rrep_sn } \rightarrow) onl \\Gamma_{\text{AODV}} (\lambda(\xi, 1).
1 \in \{\text{PAodv-:4,PAodv-:5} \cup \{\text{PRreq-:n}|n. \text{True}\} \rightarrow 1 \leq \text{osn } \xi)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp

lemma osn_rreq':
"\paodv i \mid (\text{recvmsg } \lambda m. \text{rreq_rrep_sn } m \land \text{msg_zhops } m) \rightarrow) onl \\Gamma_{\text{AODV}} (\lambda(\xi, 1).
1 \in \{\text{PAodv-:4,PAodv-:5} \cup \{\text{PRreq-:n}|n. \text{True}\} \rightarrow 1 \leq \text{osn } \xi)"
proof (rule invariant_weakenE [OF osn_rreq])
fix a
assume "\text{recvmsg } \lambda m. \text{rreq_rrep_sn } m \land \text{msg_zhops } m) a"
thus "\text{recvmsg } \text{rreq_rrep_sn } a"
by (cases a) simp_all
qed

lemma dsn_rrep:
"\paodv i \mid (\text{recvmsg } \text{rreq_rrep_sn } \rightarrow) onl \\Gamma_{\text{AODV}} (\lambda(\xi, 1).
1 \in \{\text{PAodv-:6,PAodv-:7} \cup \{\text{PRrep-:n}|n. \text{True}\} \rightarrow 1 \leq \text{dsn } \xi)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]) clarsimp
lemma dsn_rrep':
"paodv i |= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl Γ_AODV (λ(ξ, 1).
1 ∈ {PAdv-:6, PAdv-:7} ∪ {PRrep-:n|n. True} → 1 ≤ dsn ξ)"

proof (rule invariant_weakenE [OF dsn_rrep])
fix a
assume "recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a"
thus "recvmsg rreq_rrep_sn a"
by (cases a) simp_all
qed

lemma hop_count_zero_oip_dip_sip':
"paodv i |= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl Γ_AODV (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)))"

proof (rule invariant_weakenE [OF hop_count_zero_oip_dip_sip])
fix a
assume "recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a"
thus "recvmsg msg_zhops a"
by (cases a) simp_all
qed

Proposition 7.12
lemma zero_seq_unk_hops_one':
"paodv i |= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl Γ_AODV (λ(ξ, l).
∀ dip ∈ kD(rt ξ). (sqn (rt ξ) dip = 0 −→ sqnf (rt ξ) dip = unk) ∧ (sqnf (rt ξ) dip = unk −→ the (dhops (rt ξ) dip) = Suc 0) ∧
(the (dhops (rt ξ) dip) = Suc 0 −→ the (nhop (rt ξ) dip) = dip)"

proof -
{ fix dip and ξ :: 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!: bepec)
} 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 "\( \forall dip \in kD(rt). \\
(sqn rt dip = 0 \longrightarrow \pi_3(\text{the (rt dip)}) = unk) \land \\
(\pi_3(\text{the (rt dip)}) = unk \longrightarrow \text{(dhops rt dip) = Suc 0}) \land \\
(\text{(dhops rt dip) = Suc 0} \longrightarrow \text{(nhop rt dip) = 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 (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 "\( \forall dip \in kD rt. \\
(sqn rt dip = 0 \longrightarrow \pi_3(\text{the (rt dip)}) = unk) \land \\
(\pi_3(\text{the (rt dip)}) = unk \longrightarrow \text{(dhops rt dip) = Suc 0}) \land \\
(\text{(dhops rt dip) = Suc 0} \longrightarrow \text{(nhop rt dip) = dip})"
\}
hence "\( \forall dip \in kD rt. \\
(\text{sqn (update rt sip (0, unk, val, Suc 0, sip, {})) dip = 0} \longrightarrow 
\pi_3 (\text{the (update rt sip (0, unk, val, Suc 0, sip, {})) dip}) = unk) \land \\
(\pi_3 (\text{the (update rt sip (0, unk, val, Suc 0, sip, {})) dip}) = unk \longrightarrow 
\text{(dhops (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = Suc 0}) \land \\
(\text{(dhops (update rt sip (0, unk, val, Suc 0, sip, {})) dip) = Suc 0} \longrightarrow 
\text{(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]: "\( \forall sip rt. \\
\pi_3 (\text{the (update rt sip (0, unk, val, Suc 0, sip, {})) sip}) = unk \longrightarrow 
\text{(dhops (update rt sip (0, unk, val, Suc 0, sip, {})) sip) = Suc 0}" 
by (rule update_cases) simp_all

have prreq_ok6 [simp]: "\( \forall sip rt. \\
\text{sqn (update rt sip (0, unk, val, Suc 0, sip, {})) sip = 0} \longrightarrow 
\pi_3 (\text{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_TT [OF aodv_wf addpreRT_welldefined] 
onl_invariant_sterms [OF aodv_wf hop_count_zero_oip_dip_sip'] 
seq_step_invariant_sterms_TT [OF sqns_increase aodv_wf aodv_trans] 
onl_invariant_sterms [OF aodv_wf osn_rreq'] 
onl_invariant_sterms [OF aodv_wf dsn_rrep']) clarsimp+

qed

lemma zero_seq_unk_hops_one:
"paodv i ||= \{ \text{recvmsg (\( \lambda m. rreq_rrep_sn m \land msg_zhops m \))} \rightarrow onl \Gamma_{AODV} (\lambda (\xi, \_). \\
\forall dip \in kD rt. \text{(sqn (rt dip) dip = 0} \longrightarrow \text{(sqnf (rt dip) dip = unk} \\
\land \text{the (dhops (rt dip) dip) = 1} \\
\land \text{the (nhop (rt dip) dip) = dip}))\)\}
by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

lemma kD_unk_or_atleast_one:
"paodv i ||= \{ \text{recvmsg rreq_rrep_sn} \rightarrow onl \Gamma_{AODV} (\lambda (\xi, 1). \\
\forall dip \in kD rt. \text{\( \pi_3 (\text{the (rt dip)}) = unk \lor 1 \leq \pi_2 (\text{the (rt dip)})\)\})\}
proof -
{ fix sip rt dns1 dns2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhip1 nhip2 pre1 pre2 
assume "dsk1 = unk \lor Suc 0 \leq dns2"
hence "\( \pi_3 (\text{the (update rt sip (dns1, dsk1, flag1, hops1, nhip1, pre1) sip)}) = unk \lor 
Suc 0 \leq sqn (\text{update rt sip (dns2, dsk2, flag2, hops2, nhip2, pre2) sip})\)\}
}
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: "\(\forall dip \in kD(rt).\) \(\pi_3(\text{update rt sip (dsn1, dsk1, flag1, hops1, nhip1, pre1) dip}) = unk \lor 0 \leq \text{sqn rt dip}\)" 
  have "\(\forall dip \in kD(rt).\) \(\pi_3(\text{update rt sip (dsn2, dsk2, flag2, hops2, nhip2, pre2) dip}) = unk \lor 0 \leq \text{sqn rt dip}\)" 
(is "\(\forall dip \in kD(rt).\) ?prop 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 dests 
  assume *: "\(\forall ip \in \text{dom(dests)}.\) \(ip \in kD(rt) \land \text{sqn rt ip} \leq \text{the (dests ip)}\)" 
  and **: "\(\forall ip \in kD(rt).\) \(\pi_3(\text{invalidate rt dests} dip) = unk \lor 0 \leq \text{sqn rt dip}\)" 
  have "\(\forall dip \in kD(rt).\) \(\pi_3(\text{invalidate rt dests} dip) = unk \lor 0 \leq \text{sqn rt dip}\)" proof 
    fix dip 
    assume "dip \in kD(rt)" 
    with ** have "\(\pi_3(\text{invalidate rt dests} dip) = unk \lor 0 \leq \text{sqn rt dip}\)" .. 
    thus "\(\pi_3(\text{invalidate rt dests} dip) = unk \lor 0 \leq \text{sqn rt dip}\)" proof 
      assume "\(\pi_3(\text{invalidate rt dests} dip) = unk\)" thus ?thesis .. 
    next 
      assume "\(\text{sqn rt dip} \leq 0\)" 
      have "\(\text{sqn rt dip} \leq \text{the (dests dip)}\)" by simp 
      with \(\text{sqn rt dip} \leq 0\) have "\(\text{sqn rt dip} \leq \text{the (dests dip)}\)" by simp 
      with \(dip \in \text{dom(dests)}\) \(dip \in kD(rt)\) [THEN kd_Some] show ?thesis 
      unfolding invalidate_def sqn_def by auto 
    next 
      assume "dip \in \text{dom(dests)}" 
      with \(\text{sqn rt dip} \leq 0\) \(dip \in kD(rt)\) [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 aodv_wf addpreRT_welldefined] 
onl_invariant_sterms [OF aodv_wf dests_biggerRT_welldefined] 
  [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) 
proof 
  fix dip 
  assume "dip \in kD(rt)" 
  with ** have "\(\pi_3(\text{invalidate rt dests} dip) = unk \lor 0 \leq \text{sqn rt dip}\)" .. 
  thus "\(\pi_3(\text{invalidate rt dests} dip) = unk \lor 0 \leq \text{sqn rt dip}\)" proof 
    assume "\(\pi_3(\text{invalidate rt dests} dip) = unk\)" thus ?thesis .. 
  next 
    assume "\(\text{sqn rt dip} \leq 0\)" 
    have "\(\text{sqn rt dip} \leq \text{the (dests dip)}\)" by simp 
    with \(\text{sqn rt dip} \leq 0\) have "\(\text{sqn rt dip} \leq \text{the (dests dip)}\)" by simp 
    with \(dip \in \text{dom(dests)}\) \(dip \in kD(rt)\) [THEN kd_Some] show ?thesis 
    unfolding invalidate_def sqn_def by auto 
  next 
    assume "dip \in \text{dom(dests)}" 
    with \(\text{sqn rt dip} \leq 0\) \(dip \in kD(rt)\) [THEN kd_Some] show ?thesis 
    unfolding invalidate_def sqn_def by auto 
  qed 
  thus ?thesis by (rule disjI2) 
  qed 
  qed 
}
Proposition 7.13

**Lemma**: 
\[ \text{paodv} \models (\text{recvmsg} \ \text{req}\_\text{rep}\_\text{sn} \rightarrow \text{onll} \ \Gamma_{AODV} \ (\lambda(\_\_\_, \text{a}, \_\_\_.) \ \text{anycast} \ \text{req}\_\text{rep}\_\text{sn} \ \text{a})] \\
**Proof**: 
- have \( \text{sqnf\_kno} \): \( \text{paodv} \models \text{onll} \ \Gamma_{AODV} \ (\lambda(\xi, 1)). \)
  
  \( (1 \in \{\text{PRreq-:18..PRreq-:20}\} \rightarrow \text{sqnf} \ (\text{rt} \ \xi) \ (\text{dip} \ \xi) = \text{kno})\)

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

  show \( \theta\)thesis

  by (inv\_ctems inv add: onll\_invariant\_sterms\_TT [OF aodv\_wf addpreRT\_welldefined]
  onll\_invariant\_sterms [OF aodv\_wf sequence\_number\_one\_or\_bigger
  THEN invariant\_restrict\_inD])

  onll\_invariant\_sterms [OF aodv\_wf kD\_unk\_or\_atleast\_one]

  onll\_invariant\_sterms\_TT [OF aodv\_wf sqnf\_kno]

  onll\_invariant\_sterms [OF aodv\_wf osn\_req]

  onll\_invariant\_sterms [OF aodv\_wf dsn\_rep])

  (auto simp: proj2_eq_sqn)

  qed

Proposition 7.14

**Lemma**: 
\[ \text{paodv} \models \text{onll} \ \Gamma_{AODV} \ (\lambda(\xi, 1), \_\_\_, \_\_\_.) \ \text{anycast} \ \text{req}\_\text{rep}\_\text{fresh} \ (\text{rt} \ \xi) \ \text{a})] \\
**Proof**: 
- have \( \text{rreq\_oip} \): \( \text{paodv} \models \text{onll} \ \Gamma_{AODV} \ (\lambda(\xi, 1)). \)
  
  \( (1 \in \{\text{PRreq-:3..PRreq-:9}\} \cup \{\text{PRreq-:17, PRreq-:30, PRreq-:32}\}
  \rightarrow \ oip \ \xi \in \ kD(\text{rt} \ \xi)

  \& (\text{sqn} \ (\text{rt} \ \xi) \ (\text{oip} \ \xi) > (\text{osn} \ \xi))

  \lor (\text{sqn} \ (\text{rt} \ \xi) \ (\text{oip} \ \xi) = (\text{osn} \ \xi)

  \& (\text{the} \ (\text{dhops} \ (\text{rt} \ \xi) \ (\text{oip} \ \xi)) \leq \text{Suc} \ (\text{hops} \ \xi)

  \& (\text{the} \ (\text{flag} \ (\text{rt} \ \xi) \ (\text{oip} \ \xi) = \text{val}))))\)

  unfolding \text{update\_def} by (clarsimp split: option.split)

  (metis linorder\_neq\_E\_nat not\_less)

  qed

have \( \text{rrep\_prrep} \): \( \text{paodv} \models \text{onll} \ \Gamma_{AODV} \ (\lambda(\xi, 1)). \)

\( (1 \in \{\text{PRrep-:2..PRrep-:7}\} \rightarrow \ (\text{dip} \ \xi \in \ kD(\text{rt} \ \xi)

\& (\text{sqn} \ (\text{rt} \ \xi) \ (\text{dip} \ \xi) = \text{dsn} \ \xi)

\& (\text{the} \ (\text{dhops} \ (\text{rt} \ \xi) \ (\text{dip} \ \xi)) = \text{Suc} \ (\text{hops} \ \xi)

\& (\text{the} \ (\text{flag} \ (\text{rt} \ \xi) \ (\text{dip} \ \xi) = \text{val})

\& (\text{the} \ (\text{nhop} \ (\text{rt} \ \xi) \ (\text{dip} \ \xi)) \in \ kD (\text{rt} \ \xi))))\)

by (inv\_ctems inv add: onll\_invariant\_sterms [OF aodv\_wf rrep\_1\_update\_changes]

onll\_invariant\_sterms [OF aodv\_wf sip\_in\_kD])

have \( \text{rreq\_oip\_kD} \): \( \text{paodv} \models \text{onll} \ \Gamma_{AODV} \ (\lambda(\xi, 1)). \)

\( (1 \in \{\text{PRreq-:3..PRreq-:28}\} \rightarrow \ oip \ \xi \in \ kD(\text{rt} \ \xi))))\)

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

have \( \text{rreq\_dip\_kD\_oip\_sqn} \): \( \text{paodv} \models \text{onll} \ \Gamma_{AODV} \ (\lambda(\xi, 1)). \)

\( (1 \in \{\text{PRreq-:18..PRreq-:21}\}

\rightarrow \ (\text{dip} \ \xi \in \ kD(\text{rt} \ \xi)

\& (\text{sqn} \ (\text{rt} \ \xi) \ (\text{oip} \ \xi) > (\text{osn} \ \xi))

\lor (\text{sqn} \ (\text{rt} \ \xi) \ (\text{oip} \ \xi) = (\text{osn} \ \xi)

\& (\text{the} \ (\text{dhops} \ (\text{rt} \ \xi) \ (\text{oip} \ \xi)) \leq \text{Suc} \ (\text{hops} \ \xi)

\& (\text{the} \ (\text{flag} \ (\text{rt} \ \xi) \ (\text{oip} \ \xi) = \text{val}))))\)
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_oip]
onl_invariant_sterms [OF aodv_wf addpreRT_welldefined])

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]
onl_invariant_sterms [OF aodv_wf rreq_oip_kD]
onl_invariant_sterms [OF aodv_wf rreq_dip_kD_oip_sqn])
qed

Proposition 7.15

lemma rerr_invalid_any_step_invariant:
"paodv i ||=A onll Γ_AODV (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a)"
proof -
  have dests_inv: "paodv i ||=A onll Γ_AODV (λ((ξ, _), a, _). anycast (rerr_invalid (rt ξ)) a)
  (∀ip∈dom(dests ξ). ip∈vD(rt ξ)))
∧ (1 ∈ {PAodv=15, PPkt=7, PRreq=11, PRreq=24, PRrep=10, PRerr=1})
  ∨ (1 ∈ {PAodv=16..PAodv=19})
  ∨ (1 ∈ {PPkt=8..PPkt=11})
  ∨ (1 ∈ {PRreq=12..PRreq=15})
  ∨ (1 ∈ {PRreq=25..PRreq=28})
  ∨ (1 ∈ {PRrep=11..PRrep=14})
  ∨ (1 ∈ {PRrep=2..PRrep=5})
  → (∀ip∈dom(dests ξ). ip∈D(rt ξ) ∧ (dests ξ ip) = inc (sqn (rt ξ) ip))
∧ (1 = PPkt=14 → dip ξ∈D(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 aodv_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 ↦ 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:
"paodv i ||=A onll Γ_AODV (λ(ξ, 1). (1 ∈ {PAodv=15, PPkt=7, PRreq=11, PRreq=24, PRrep=10, PRerr=1})
  → (∀ip∈dom(dests ξ). ip∈vD(rt ξ)))
∧ (1 ∈ {PAodv=16..PAodv=19})
  ∨ (1 ∈ {PPkt=8..PPkt=11})
  ∨ (1 ∈ {PRreq=12..PRreq=15})
  ∨ (1 ∈ {PRreq=25..PRreq=28})
  ∨ (1 ∈ {PRrep=11..PRrep=14})
  ∨ (1 ∈ {PRrep=2..PRrep=5})
  → (∀ip∈dom(dests ξ). ip∈D(rt ξ) ∧ (dests ξ ip) = inc (sqn (rt ξ) ip))
∧ (1 = PPkt=14 → dip ξ∈D(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 aodv_wf dests_vD_inc_sqn])
qed

Proposition 7.27

lemma route_tables_fresher:
"paodv i ||=A (recvmsg rreq_rrep_sn → onll Γ_AODV (λ((ξ, _), _, (ξ', _)).
  ∨ dip∈kD(rt ξ). rt ξ ⊆ dip rt ξ'))"
proof (inv_cterms inv add:
onl_invariant_sterms [OF aodv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]]
onl_invariant_sterms [OF aodv_wf osn_rreq]
onl_invariant_sterms [OF aodv_wf dsn_rrep]
onl_invariant_sterms [OF aodv_wf addpreRT_welldefined [THEN invariant_restrict_inD]]

380
4.7 The quality increases predicate

theory D_Quality_Increases
imports D_Aodv_Predicates D_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 (σ nhip)"
and qinc: "quality_increases (σ nhip) (σ' nhip)"
and "dip∈kD(rt (σ nhip))"
shows "rt (σ i) ⊏ dip (σ' nhip)"
proof -
from qinc have "rt (σ nhip) ⊑ dip (σ' nhip)" using ⟨dip∈kD(rt (σ nhip))⟩
by auto
with ⟨rt (σ i) ⊏ dip (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'⟩ have "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 "sqn (rt (σ sip)) ip ≥ 1" by simp
with ⟨sqn (rt (σ' sip)) ip ≥ the (dhops (rt (σ' sip)) ip)⟩ have "nsqn (rt (σ' sip)) ip > sn" by simp
thus ?thesis ..

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)
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 rreqid dip dsn dsk oip osn sip handled ⇒
  (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))))
| Rrep hops dip dsn dipc dsnc sipc ⇒
  (dsn ≥ 1 ∧ (sip ≠ dipc → dipc ∈ kD(rt (σ sip))) ∧ nsqn (rt (σ sip)) dipc ≥ dsn
  ∧ (nsqn (rt (σ sip)) dipc = dsn
  → (hops ≥ the (dhops (rt (σ sip))) dipc)
  ∨ the (flag (rt (σ sip)) dipc) = inv))))
| Rerr dests sipc ⇒
  (∀ ripc ∈ dom(dests). (ripc ∈ kD(rt (σ sipc)) ∧ the (dests ripc) = 1 ≤ nsqn (rt (σ sipc)) ripc))
| _ ⇒ True"

lemma msg_fresh [simp]: "∀ hops rreqid dip dsn dsk oip osn sip handled.
  msg_fresh σ (Rreq hops rreqid dip dsn dsk oip osn sip handled) =
  (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)))"
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 ..
  qed
  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 \( \text{rt} = "\text{rt } (\sigma \text{ sip})" \)

from assms have \(*\): "(\(\forall \text{rip} \in \text{dom(dests)}. (\text{rip} \in \text{ID}(\text{rt } (\sigma \text{ sip}))) \land \text{the } (\text{dests rip}) = \text{sqn } (\text{rt } (\sigma \text{ sip})) \text{ rip})""

proof

from this(2) have "(\(\forall \text{rip} \in \text{dom(dests)}. (\text{rip} \in \text{KD}(\text{rt } (\sigma \text{ sip}))) \land \text{the } (\text{dests rip}) - 1 \leq \text{nsqn } (\text{rt } (\sigma \text{ sip})) \text{ rip})".

with \(\langle \text{rip} \in \text{KD}(\text{rt } (\sigma \text{ sip})) \rangle\)

thus "msg_fresh \(\sigma \ ?\text{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 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 \(\sigma \ m""

then have "osn \(\geq 1" \) and "sip = oip \lor (\text{oip} \in \text{KD}(\text{rt } (\sigma \text{ sip})) \land \text{osn} \leq \text{nsqn } (\text{rt } (\sigma \text{ sip})) \text{ oip} \land (\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} \lor \text{the } (\text{flag } (\text{rt } (\sigma \text{ sip})) \text{ oip} = \text{inv}))""

by auto

from this(2) show ?thesis

proof

assume "sip = oip" with \(\langle \text{osn} \geq 1 \rangle\) show ?thesis by simp

next

assume "oip \in \text{KD}(\text{rt } (\sigma \text{ sip})) \land \text{osn} \leq \text{nsqn } (\text{rt } (\sigma \text{ sip})) \text{ oip} \land (\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} \lor \text{the } (\text{flag } (\text{rt } (\sigma \text{ sip})) \text{ oip} = \text{inv}))""

moreover from qinc have "quality_increases (\(\sigma \text{ sip}) (\sigma' \text{ sip})" ..

ultimately have "oip \in \text{KD}(\text{rt } (\sigma' \text{ sip})) \land \text{osn} \leq \text{nsqn } (\text{rt } (\sigma' \text{ sip})) \text{ oip} \land (\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} \lor \text{the } (\text{flag } (\text{rt } (\sigma' \text{ sip})) \text{ oip} = \text{inv}))""

using \(\langle \text{osn} \geq 1 \rangle\) by (rule quality_increases_rreq_rrep_props [rotated 2])

with \(\langle \text{osn} \geq 1 \rangle\) 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 \lor (\text{dip} \in \text{KD}(\text{rt } (\sigma \text{ sip})) \land \text{dsn} \leq \text{nsqn } (\text{rt } (\sigma \text{ sip})) \text{ dip} \land (\text{nsqn } (\text{rt } (\sigma \text{ sip})) \text{ dip} = \text{dsn} \rightarrow (\text{the } (\text{dhops } (\text{rt } (\sigma \text{ sip})) \text{ dip}) \leq \text{hops} \lor \text{the } (\text{flag } (\text{rt } (\sigma \text{ sip})) \text{ dip} = \text{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_Open
imports D_Oadv "./././AWN/OAWN_SOS_Labels" "./././AWN/OAWN_Convert"
begin
Definitions for stating and proving global network properties over individual processes.
definition σ_AODV' :: "((ip ⇒ state) × ((state, msg, pseqp, pseqp label) seqp)) set"
where "σ_AODV' ≡ {(λi. aodv_init i, Γ_AODV PAodv)}"
abbreviation opaodv :: "ip ⇒ ((ip ⇒ state) × (state, msg, pseqp, pseqp label) seqp, msg seq_action) automaton"
where "opaodv i ≡ (| init = σ_AODV', trans = oseqp_sos Γ_AODV i |)"
lemma initiali_aodv [intro!, simp]: "initiali i (init (opaodv i)) (init (paodv i))"
unfolding σ_AODV'_def σ_AODV'_def by rule simp_all
lemma oaodv_control_within [simp]: "control_within Γ_AODV (init (opaodv i))"
unfolding σ_AODV'_def by (rule control_withinI) (auto simp del: Γ_AODV.simps)
lemma σ_AODV'_labels [simp]: "(σ, p) ∈ σ_AODV' ⇒ labels Γ_AODV p = {PAodv-:0}"
unfolding σ_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/OAWN_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 (orecmd msg Q) \sigma \sigma' a"
and weakenP: "\forall \sigma m. P \sigma m \implies P' \sigma m"
and weakenQ: "\forall \sigma m. Q \sigma m \implies Q' \sigma m"
shows "otherwith P' I (orecmd msg 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 "orecmd msg Q a" by auto
thus "orecmd msg Q' a" by rule (erule weakenQ)
qed

lemma orecevd_msg_inv:
assumes other: "\forall \sigma \sigma'. m \parallel P \sigma m; other Q \{i\} \sigma \sigma' \parallel \implies P' \sigma m" 
and local: "\forall \sigma m. P \sigma m \implies P (\sigma (i := \sigma i (msg := m))) m"
shows "opaodv i \parallel (otherwith Q \{i\} (orecmdmsg P), other Q \{i\} \to) onl \Gamma_{AODV} (\lambda(\sigma, l). \ l \in \{PAodv-:1\} \implies P \sigma (msg (\sigma i)) )"
proof (inv_cterms, intro impl)
fix \sigma \sigma' l
assume "l = PAodv-:1 \implies 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

dariusz@jacobs-university.de
next
  fix σ σ' msg
  assume "otherwith Q {i} (orecvmmsg 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

(Equivalent to) Proposition 7.27

lemma local_quality_increases:
  "paodv i |=A (recmsg rreq_rrep_sn →) onll ΓAODV (λ((ξ, _), _, (ξ', _)). quality_increases ξ ξ')"
proof (rule step_invariantI)
  fix s a s'
  assume sr: "s ∈ reachable (paodv i) (recvmsg rreq_rrep_sn)"
  and tr: "(s, a, s') ∈ trans (paodv i)"
  and rm: "recvmsg rreq_rrep_sn a"
  from sr have srTT: "s ∈ reachable (paodv 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_aodv oaodv_trans aodv_trans,
  simplified seqll_onll_swap]

lemma equality_increases:
  "opaodv i |=A {otherwith quality_increases {i} (orecvmmsg λ_. rreq_rrep_sn)},
  other quality_increases {i} (→)
  onll ΓAODV (λ((ξ, _), _, (ξ', _)). ∀ j. quality_increases (σ j) (σ' j))"
  (is "|=A (?S, _ →) _")
proof (rule onll_ostep_invariantI, simp)
  fix σ p 1 a σ' p' 1'
  assume or: "((σ, p) ∈ orachable (opaodv i) ?S (other quality_increases {i})),
  other quality_increases {i} (→)
  onll ΓAODV (λ((ξ, _), _, (ξ', _)). ∀ j. quality_increases (σ j) (σ' j))"
  and 11: "1 ∈ labels ΓAODV p"
  and ?S σ σ' a"
  and tr: "((σ, p), a, (σ', p')) ∈ oseqp_sos ΓAODV i"
  and 11: "1' ∈ labels ΓAODV p'"
  from this(1-3) have "orecvmmsg (λ_. rreq_rrep_sn) σ a" 
  by (auto dest!: orachable_weakenE [where QS="act (recvmmsg rreq_rrep_sn)"
  and QU="other quality_increases {i}"]
  otherwith_actionD)
  with or have orw: "((σ, p) ∈ orachable (opaodv i) (act (recvmmsg rreq_rrep_sn)))
  (other quality_increases {i})"
  by - (erule orachable_weakenE, auto)
with tr 11 l' and (recvmsg (\_. req 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 \(i \neq \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:
"opaodv 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 a \sigma' p a'
assume or: "((\sigma, p) \in oreachable (opaodv i) (act (recvmsg rreq rrep sn)) (other A \{i\})"
and recv: "act (recvmsg rreq rrep sn) \sigma a \sigma' p a'"
obtain l l' where "l \in labels \Gamma_{AODV} p" and "l' \in labels \Gamma_{AODV} p'"
by (metis aodv_ex_label)
from \((\sigma, p), a, (\sigma', p')) \in oseqp_sos \Gamma_{AODV} i"
have tr: "((\sigma, p), a, (\sigma', p')) \in trans (opaodv i)"
by simp
have "anycast (rreq_rrep_fresh (rt (\sigma i))) a"
proof
have "opaodv i \models_A (act (recvmsg rreq rrep sn), other A \{i\} \rightarrow)
onll \Gamma_{AODV} (\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_aodv, 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 \l \in labels \Gamma_{AODV} p and \l' \in labels \Gamma_{AODV} p' by auto
qed
moreover have "anycast (rerr_invalid (rt (\sigma i))) a"
proof
have "opaodv 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_aodv, 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 \l \in labels \Gamma_{AODV} p and \l' \in 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 aodv_trans, simplified seqll_onll_swap]])
thus \thesis
using \l \in labels \Gamma_{AODV} p and \l' \in labels \Gamma_{AODV} p' by auto
qed
moreover have "anycast (\lambda m. not_Pkt m \rightarrow msg_sender m = i) a"
proof
have "opaodv 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 rreq_rrep_sn_any_step_invariant initiali_aodv oadv_trans aodv_trans, simplified seqll_onll_swap]])
thus \thesis
using \l \in labels \Gamma_{AODV} p and \l' \in labels \Gamma_{AODV} p' by 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
by auto
qed

lemma oreceived_rreq_rrep_nsqn_fresh_inv:
"opaodv i |= (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i} →)
onl Γ_AODV (λ((σ, _), a, _. anycast (msg_fresh σ) a) ((σ, p), a, (σ′, p')))"
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 Γ_AODV (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} →)
onll Γ_AODV (seql 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: seql_onl_swap)

lemma rreq_sip:
"opaodv i |= (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other quality_increases {i} →)
onll Γ_AODV (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 oadv_trans]
onll oinvariant_sterms [OF aodv_wf oreceived_rreq_rrep_nsqn_fresh_inv]
onll oinvariant_sterms [OF aodv_wf oosn_rreq]
(auto simp: seqlsimp)
\begin{verbatim}
fix \sigma\,\epsilon\,p\,l
assume "(\sigma, p) \in \text{reachable (opaodv i) }?S\,?U"
and "1 \in \text{labels } \Gamma_{AODV} p"
and pre:
"(1 = PAodv-:4 \lor 1 = PAodv-:5 \lor 1 = PRreq-:0 \lor 1 = PRreq-:2) \land \text{sip } (\sigma i) \neq \text{oip } (\sigma i)
\rightarrow 
\text{oip } (\sigma i) \in \text{kD}(rt (\sigma (\text{sip } (\sigma i)))
\land \text{osn } (\sigma i) \leq \text{nsqn}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i))
\land (\text{nsqn}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i)) = \text{osn } (\sigma i)
\rightarrow \text{the } (\text{dhops}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i))) \leq \text{hops } (\sigma i)
\lor \text{the } (\text{flag}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i))) = \text{inv})"
and "\text{other_quality_increases } \{i\} \text{ }\sigma'"
and hyp: "(1 = PAodv-:4 \lor 1 = PAodv-:5 \lor 1 = PRreq-:0 \lor 1 = PRreq-:2) \land \text{sip } (\sigma i) \neq \text{oip } (\sigma i)"
(is "?labels \land \text{sip } (\sigma i) \neq \text{oip } (\sigma i)"
from(4) have "\sigma i = \sigma i"
with hyp have hyp": "?labels \land \text{sip } (\sigma i) \neq \text{oip } (\sigma i)" by simp
show "\text{oip } (\sigma i) \in \text{kD}(rt (\sigma (\text{sip } (\sigma i))))
\land \text{osn } (\sigma i) \leq \text{nsqn}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i))
\land (\text{nsqn}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i)) = \text{osn } (\sigma i)
\rightarrow \text{the } (\text{dhops}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i))) \leq \text{hops } (\sigma i)
\lor \text{the } (\text{flag}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i))) = \text{inv})"
proof (cases "\text{sip } (\sigma i) = i")
assume "\text{sip } (\sigma i) \neq i"
from "\text{other_quality_increases } \{i\} \text{ }\sigma'"
have "\text{quality_increases } (\sigma (\text{sip } (\sigma i)))(\sigma' (\text{sip } (\sigma i)))"
by (rule otherE) (clarsimp simp: (\sigma i) \neq i)
moreover from "(\sigma, p) \in \text{reachable (opaodv i) }?S\,?U\land 1 \in \text{labels } \Gamma_{AODV} p" and hyp
have "1 \leq \text{osn } (\sigma i)"
by (auto dest!: onl_oninvariant_weakenD[OF osn_rreq]
simp add: seqlsimp (\sigma i = i))
moreover from :\text{sip } (\sigma i) \neq i) hyp' and pre
have "\text{oip } (\sigma i) \in \text{kD}(rt (\sigma (\text{sip } (\sigma i))))
\land \text{osn } (\sigma i) \leq \text{nsqn}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i))
\land (\text{nsqn}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i)) = \text{osn } (\sigma i)
\rightarrow \text{the } (\text{dhops}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i))) \leq \text{hops } (\sigma i)
\lor \text{the } (\text{flag}(rt (\sigma (\text{sip } (\sigma i)))) (\text{oip } (\sigma i))) = \text{inv})"
by (auto simp: (\sigma i = i))
ultimately show ?thesis
by (rule quality_increases_rreq_rrep_props)
next
assume "\text{sip } (\sigma i) = i" thus ?thesis
using (\sigma i = i) hyp and pre by auto
qed
qd
qd (auto elim!: quality_increases_rreq_rrep_props')

lemma odsn_rrep:
"opaodv i \mid (\text{otherwith quality_increases } \{i\} \text{ (orecvmsg msg_fresh)}, \text{ other quality_increases } \{i\} \rightarrow)
onl \Gamma_{AODV} (\text{seql } (\lambda(\xi, 1). \text{l} \in \{\text{PAodv-:6, PAodv-:7, PRrep-:0, PRrep-:1}\} \cup \{\text{PRrep-:n\mid n. True} \rightarrow 1 \leq \text{dsn } \xi\}))"
by (rule oninvariant_weakenE[OF open_seq_invariant[OF dsn_rrep initial_i_aodv]])
(auto simp: seql_onl_swap)

lemma rrep_sip:
"opaodv i \mid (\text{otherwith quality_increases } \{i\} \text{ (orecvmsg msg_fresh)}, \text{ other quality_increases } \{i\} \rightarrow)
onl \Gamma_{AODV} (\lambda(\sigma, 1).
(\text{l} \in \{\text{PAodv-:6, PAodv-:7, PRrep-:0, PRrep-:1}\} \land \text{sip } (\sigma i) \neq \text{dip } (\sigma i))
\rightarrow \text{dip } (\sigma i) \in \text{kD}(rt (\sigma (\text{sip } (\sigma i)))
\land \text{nsqn}(rt (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i)) \geq \text{dsn } (\sigma i)
\land (\text{nsqn}(rt (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i)) = \text{dsn } (\sigma i)
\rightarrow \text{the } (\text{dhop}(rt (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i))) \geq \text{the } (\text{flag}(rt (\sigma (\text{sip } (\sigma i)))) (\text{dip } (\sigma i))) = \text{inv})))"
(is "\mid (?S, ?U \rightarrow _")
proof (inv_cterms inv add: oseq_step_invariant_terms[OF equality_increases_nsqn_fresh aodv_wf]
\end{verbatim}
lemma rerr_sip:
(rerr_sip: \( \text{auto simp add: seqlsimp elim!: quality_increases_rreq_rrep_props') }\)
qed

"opaodv i \(\equiv\) (otherwith quality_increases \{i\} \text{ (orecvmsg msg_fresh)},
other_quality_increases \{i\} \rightarrow\)
onl \(\Gamma_{AODV} (\lambda(s, l). \)
\(1 \in \{PAodv-:8, PAodv-:9, PRrep-:0, PRreq-:1\} \rightarrow \)
\(\forall \text{ ripc} \in \text{dom(dests (s i)). ripc} \in kD (rt (s (sip (s i)))) \land \)
\(\text{the (dests (s i) ripc) - 1} \leq \text{nsqn (rt (s (sip (s i)))) ripc})\)
(is "_ _ \(\equiv\) (S, ?U \rightarrow \_ _)\)
proof -
\{ fix dests rip sip rsn and \(s_\sigma\) :: "ip \(\equiv\) state"
assume qinc: "\(\forall j. \text{quality_increases (s j) (s' j)}\)"
and *: "\(\forall \text{ ripc} \in \text{dom dests. ripc} \in kD (rt (s sip)) \land \)
\(\text{the (dests rip) - 1} \leq \text{nsqn (rt (s sip)) rip})\)
and "dests rip \(\equiv\) Some rsn"
from this(3) have "rip \(\equiv\) dom dests" by auto
with * and \(\text{dests rip} \text{ = Some rsn}\) have "rip \(\in\) kD (rt (s 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)) \land rsn - 1 \leq nsqn (rt (\sigma' \, sip)) \, rip"
proof
from \langle rip \in kD(rt (\sigma \, sip)) \rangle \land \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 \land \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

have "rip \in kD(rt (\sigma' \, sip))" ..

next
from \langle rip \in kD(rt (\sigma \, sip)) \rangle \land \langle quality_increases (\sigma \, sip) (\sigma' \, sip) \rangle
have "nsqn (rt (\sigma \, sip)) \, rip \leq nsqn (rt (\sigma' \, sip)) \, rip"

proof (intro conjI)

from \langle rip \in kD(rt (\sigma \, sip)) \rangle \land \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

\text{lemma prerr_guard: "paodv \, i \models onl \Gamma_{AODV} (\lambda(\xi, \_). \, (1 = PRe1; -1 \\
\rightarrow (\forall ip \in \text{dom}(\text{dests} \, \xi)). \, ip \in \text{vD}(rt \, \xi) \\
\land \, \text{the} \, (\text{nhop} \, (rt \, \xi) \, ip) = sip \xi \\
\land \, \text{sqn} \, (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_aodv oadv_trans aodv_trans, 
simplified seql_onl_swap, 
THEN oinvariant_anyact]

lemmas odests_vD_inc_sqn =
open_seq_invariant [OF dests_vD_inc_sqn initiali_aodv oadv_trans aodv_trans, 
simplified seql_onl_swap, 
THEN oinvariant_anyact]

lemmas oprerr_guard =
open_seq_invariant [OF prerr_guard initiali_aodv oadv_trans aodv_trans, 
simplified seql_onl_swap, 
THEN oinvariant_anyact]

Proposition 7.28

\text{lemma seq_compare_next_hop':
"opaodv \, i \models (\text{otherwith quality_increases \{i\} (orecvmsg \, msg_fresh)}, 
other quality_increases \{i\} \rightarrow) \, onl \Gamma_{AODV} (\lambda(\sigma, \_). \\
\forall dip. \, \text{let} \, nhip = \text{the} \, (\text{nhop} \, (rt \, \sigma \, i)) \, dip \rightarrow \\
\text{in} \, dip \in kD(rt (\sigma \, i)) \land nhip \neq dip \rightarrow \\
\text{in} \, dip \in kD(rt (\sigma' \, nhip)) \land \text{sqn} \, (rt (\sigma \, i)) \, dip \leq \text{sqn} \, (rt (\sigma \, nhip)) \, dip)"
(is "\_ \models (\_ \, ?U \rightarrow \_ \)\)

proof -

{ fix nhip and \sigma \, \sigma' :: "ip \Rightarrow \text{state}"
assume pre: "\forall dip \in kD(rt (\sigma \, i)). \, \text{nhop} \, dip \neq dip \rightarrow \\
\text{dip} \in kD(rt (\sigma' \, (\text{nhop} \, dip))) \land \text{sqn} \, (rt (\sigma \, i)) \, dip \leq \text{sqn} \, (rt (\sigma' \, (\text{nhop} \, dip))) \, dip"
and qinc: "\forall j. \, \text{quality_increases} \, (\sigma \, j) \, (\sigma' \, j)"
have "\forall dip \in kD(rt (\sigma \, i)). \, \text{nhop} \, dip \neq dip \rightarrow \\
\text{dip} \in kD(rt (\sigma' \, (\text{nhop} \, dip))) \land \text{sqn} \, (rt (\sigma \, i)) \, dip \leq \text{sqn} \, (rt (\sigma' \, (\text{nhop} \, dip))) \, dip"
proof (intro ballI impI)
fix dip

395
assume "\(\text{dip} \in \text{kD}(\text{rt} (\sigma i))\)"
and "\(\text{nhop} \text{dip} \neq \text{dip}\)"
with pre have "\(\text{dip} \in \text{kD}(\text{rt} (\sigma (\text{nhop dip})))\)"
and "\(\text{nsqn} (\text{rt} (\sigma i)) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma (\text{nhop dip}))) \text{ dip}\)"
by auto
from qinc have qinc_nh: "\(\text{quality} \_\text{increases} (\sigma (\text{nhop dip})) (\sigma' (\text{nhop dip}))\)" ..
with \(\text{dip} \in \text{kD}(\text{rt} (\sigma (\text{nhop dip})))\) have "\(\text{dip} \in \text{kD}(\text{rt} (\sigma' (\text{nhop dip})))\)" ..
moreover have "\(\text{nsqn} (\text{rt} (\sigma i)) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop dip}))) \text{ dip}\)"
proof -
from \(\text{dip} \in \text{kD}(\text{rt} (\sigma (\text{nhop dip})))\) \(\text{qinc}\) have "\(\text{nsqn} (\text{rt} (\sigma (\text{nhop dip}))) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma (\text{nhop dip}))) \text{ dip}\)" ..
with \(\text{nsqn} (\text{rt} (\sigma i)) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma (\text{nhop dip}))) \text{ dip}\) show \(?\text{thesis}\) by simp
qed
ultimately show "\(\text{dip} \in \text{kD}(\text{rt} (\sigma' (\text{nhop dip})))\)"
and "\(\text{nsqn} (\text{rt} (\sigma i)) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop dip}))) \text{ dip}\)" ..
qed

\{ fix nhop and \(\sigma, \sigma' : \text{"ip} \Rightarrow \text{state}\) 
assume pre: "\(\forall \text{dip} \in \text{kD}(\text{rt} (\sigma i)). \text{nhop} \text{dip} \neq \text{dip} \rightarrow \text{dip} \in \text{kD}(\text{rt} (\sigma (\text{nhop dip})))\)"
and ndest: "\(\forall \text{ ripc} \in \text{dom} (\text{dests} (\sigma i)). \text{ ripc} \in \text{kD} (\text{rt} (\sigma (\text{sip} (\sigma i))))\)"
and issip: "\(\forall i\). \text{quality} \_\text{increases} (\sigma j) (\sigma' j)\)"
have "\(\forall \text{dip} \in \text{kD}(\text{rt} (\sigma i)). \text{nhop} \text{dip} \neq \text{dip} \rightarrow \text{dip} \in \text{kD} (\text{rt} (\sigma' (\text{nhop dip})))\)"
and "\(\text{nsqn} (\text{invalidate} (\text{rt} (\sigma i)) (\text{dests} (\sigma i))) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop dip}))) \text{ dip}\)"
proof (intro ballI impI)
fix dip
assume "\(\text{dip} \in \text{kD}(\text{rt} (\sigma i))\)"
and "\(\text{nhop dip} \neq \text{dip}\)"
with pre and qinc have "\(\text{dip} \in \text{kD}(\text{rt} (\sigma' (\text{nhop dip})))\)"
and "\(\text{nsqn} (\text{rt} (\sigma i)) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop dip}))) \text{ dip}\)"
by (auto dest!: basic)
have "\(\text{nsqn} (\text{invalidate} (\text{rt} (\sigma i)) (\text{dests} (\sigma i))) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop dip}))) \text{ dip}\)"
proof (cases "\(\text{dip} \in \text{dom} (\text{dests} (\sigma i))\)"
assume "\(\text{dip} \in \text{dom} (\text{dests} (\sigma i))\)"
with \(\text{dip} \in \text{kD}(\text{rt} (\sigma i))\) obtain dsn where "\(\text{dests} (\sigma i) \text{ dip} = \text{Some} \ dsn\)"
by auto
with \(\text{dip} \in \text{kD}(\text{rt} (\sigma i))\) have "\(\text{nsqn} (\text{invalidate} (\text{rt} (\sigma i)) (\text{dests} (\sigma i))) \text{ dip} = \text{dsn} - 1\)"
by (rule rule_qinc_invalidate_eq)
moreover have "\(\text{dsn} - 1 \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop dip}))) \text{ dip}\)"
proof -
from \(\text{dests} (\sigma i) \text{ dip} = \text{Some} \ \text{dsn}\) have "\(\text{the} (\text{dests} (\sigma i) \text{ dip} = \text{dsn})\)" by simp
with ndest and \(\text{dip} \in \text{dom} (\text{dests} (\sigma i))\) have "\(\text{dip} \in \text{kD} (\text{rt} (\sigma (\text{sip} (\sigma i))))\)"
"\(\text{dsn} - 1 \leq \text{nsqn} (\text{rt} (\sigma (\text{sip} (\sigma i)))) \text{ dip}\)"
by auto
moreover from issip and \(\text{dip} \in \text{dom} (\text{dests} (\sigma i))\) have "\(\text{nhop dip} = \text{sip} (\sigma i)\)" ..
ultimately have "\(\text{dip} \in \text{kD} (\text{rt} (\sigma (\text{nhop dip})))\)"
and "\(\text{dsn} - 1 \leq \text{nsqn} (\text{rt} (\sigma (\text{nhop dip}))) \text{ dip}\)" by auto
with qinc show "\(\text{dsn} - 1 \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop dip}))) \text{ dip}\)" by simp (metis kD_nsqn_quality_increases_trans)
qed
ultimately show \(?\text{thesis}\) by simp
next
assume "\(\text{dip} \notin \text{dom} (\text{dests} (\sigma i))\)"
with \(\text{dip} \in \text{kD}(\text{rt} (\sigma i))\) have "\(\text{nsqn} (\text{invalidate} (\text{rt} (\sigma i)) (\text{dests} (\sigma i))) \text{ dip} = \text{nsqn} (\text{rt} (\sigma i)) \text{ dip}\)"
by (rule rule_qinc_invalidate_other)
with \(\text{nsqn} (\text{rt} (\sigma i)) \text{ dip} \leq \text{nsqn} (\text{rt} (\sigma' (\text{nhop dip}))) \text{ dip}\) show \(?\text{thesis}\) by simp
qed

with \( \langle \text{dip} \in \text{kD} \rangle \text{(rt (\(\sigma'\) (nhop dip)))} \)

show "\(\text{dip} \in \text{kD} \langle \text{rt (\(\sigma'\) (nhop dip))} \)\)
\(\land \text{nsqn (invalidate (rt (\(\sigma\) i)) (dests (\(\sigma\) i)))} \text{ dip} \leq \text{nsqn (rt (\(\sigma'\) (nhop dip))))} \)" ..

qed

} note basic_perrr = this

{ fix \(\sigma'\) :: "ip \(\Rightarrow\) state"

assume a1: "\(\forall \text{dip} \in \text{kD} \langle \text{rt (\(\sigma\) i))} \). the (nhop (rt (\(\sigma\) i)) dip) \(\neq\) dip

\(\rightarrow\) dip \(\in\) kD(\(\sigma' (\text{the (nhop (rt (\(\sigma\) i)) dip))))\)

\(\land\) nsqn (rt (\(\sigma\) i)) dip \(\leq\) nsqn (rt (\(\sigma' (\text{the (nhop (rt (\(\sigma\) i)) dip))))\)) dip"

and a2: "\(\forall j. \text{quality_increases} (\sigma j) (\sigma' j)\)"

have "\(\forall \text{dip} \in \text{kD} (\text{rt (\(\sigma\) i))}\).

the (nhop (update (rt (\(\sigma\) i)) (sip (\(\sigma\) i)) \(\langle\) 0, unk, val, Suc 0, sip (\(\sigma\) i), {} \(\rangle\)) dip) \(\neq\) dip

\(\rightarrow\) dip \(\in\) kD(\(\sigma' (\text{the (nhop (rt (\(\sigma\) i)) dip))))\)

\(\land\) nsqn (update (rt (\(\sigma\) i)) (sip (\(\sigma\) i)) \(\langle\) 0, unk, val, Suc 0, sip (\(\sigma\) i), {} \(\rangle\)) dip

\(\leq\) nsqn (rt (\(\sigma' (\text{the (nhop (rt (\(\sigma\) i)) dip))))\)

\(\langle\) 0, unk, val, Suc 0, sip (\(\sigma\) i), {} \(\rangle\)) dip)

\)” dip" (is "\(\forall \text{dip} \in \text{kD} (\text{rt (\(\sigma\) i))}. ?P dip")

proof

fix dip

assume "dip \(\in\) kD (\text{rt (\(\sigma\) i))}"

with a1 and a2

have "\(\forall \text{dip} \in \text{kD} (\text{rt (\(\sigma\) i))}\).

the (nhop (update (rt (\(\sigma\) i)) (sip (\(\sigma\) i)) \(\langle\) 0, unk, val, Suc 0, sip (\(\sigma\) i), {} \(\rangle\)) dip) \(\neq\) dip

\(\rightarrow\) dip \(\in\) kD(\(\sigma' (\text{the (nhop (rt (\(\sigma\) i)) dip))))\)

\(\land\) nsqn (update (rt (\(\sigma\) i)) (sip (\(\sigma\) i)) \(\langle\) 0, unk, val, Suc 0, sip (\(\sigma\) i), {} \(\rangle\)) dip

\(\leq\) nsqn (rt (\(\sigma' (\text{the (nhop (rt (\(\sigma\) i)) dip))))\)

\(\langle\) 0, unk, val, Suc 0, sip (\(\sigma\) i), {} \(\rangle\)) dip)

\)” dip" (is "\(\forall \text{dip} \in \text{kD} (\text{rt (\(\sigma\) i))}. ?P dip") by - (drule(1) basic, auto)

thus "?P dip" by (cases "dip = sip (\(\sigma\) i)") auto

qed

} note nhop_update_sip = this

{ fix \(\sigma'\) oip sip osn hops

assume pre: "\(\forall \text{dip} \in \text{kD} (\text{rt (\(\sigma\) i))}. the (nhop (rt (\(\sigma\) i)) dip) \(\neq\) dip

\(\rightarrow\) dip \(\in\) kD(\(\sigma' (\text{the (nhop (rt (\(\sigma\) i)) dip))))\)

\(\land\) nsqn (rt (\(\sigma\) i)) dip \(\leq\) nsqn (rt (\(\sigma' (\text{the (nhop (rt (\(\sigma\) i)) dip))))\)) dip"

and qinc: "\(\forall j. \text{quality_increases} (\sigma j) (\sigma' j)\)"

and *: "sip \(\neq\) oip \(\rightarrow\) oip \(\in\) kD(\(\sigma \text{ sip})\)

\(\land\) osn \(\leq\) nsqn (rt (\(\sigma\) sip)) oip

\(\land\) (nsqn (rt (\(\sigma\) sip)) oip = osn

\(\rightarrow\) the (dhops (rt (\(\sigma\) sip)) oip) \(\leq\) hops

\(\lor\) the (flag (rt (\(\sigma\) sip)) oip) = inv)"

from pre and qinc

have pre': "\(\forall \text{dip} \in \text{kD} (\text{rt (\(\sigma\) i))}. the (nhop (rt (\(\sigma\) i)) dip) \(\neq\) dip

\(\rightarrow\) dip \(\in\) kD(\(\sigma' (\text{the (nhop (rt (\(\sigma\) i)) dip))))\)

\(\land\) nsqn (rt (\(\sigma\) i)) dip \(\leq\) nsqn (rt (\(\sigma' (\text{the (nhop (rt (\(\sigma\) i)) dip))))\)) dip"

by (rule basic)

have "\((\text{the (nhop (update (rt (\(\sigma\) i)) oip (osn, kno, val, Suc hops, sip, {})) oip}) \(\neq\) oip

\(\rightarrow\) oip \(\in\) kD(\(\sigma' (\text{the (nhop (update (rt (\(\sigma\) i)) oip}

\(\langle\) osn, kno, val, Suc hops, sip, {} \(\rangle\)) oip))))\)

\(\land\) nsqn (update (rt (\(\sigma\) i)) oip (osn, kno, val, Suc hops, sip, {})) oip

\(\leq\) nsqn (rt (\(\sigma' (\text{the (nhop (update (rt (\(\sigma\) i)) oip}

\(\langle\) osn, kno, val, Suc hops, sip, {} \(\rangle\)) oip)))) oip)"

(is "?\text{nhop_not_oip} \(\rightarrow\) ?\text{oip in kD} \(\land\) ?\text{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 "?\text{oip in kD} \(\land\) ?\text{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: ?\text{nhop_not_oip}

with * qinc have ?\text{oip in kD}

by (clarsimp elim!: kD_quality_increases)

397
moreover with \* pre qinc rtnot notoip have \?nsqn_le_nsqn

by simp (metis kD_nsqn_quality_increases_trans)
ultimately show \(?oip_in_kD \land \?nsqn_le_nsqn\) ...

qed

} note update1 = this

\{ fix \(\sigma\', \oip\) sip osn hops
assume pre: "\(\forall dip \in kD (rt (\sigma i)). \) the \(\langle nhop (rt (\sigma i)) dip\rangle \neq dip\)
\rightarrow dip \in kD (rt (\sigma (\langle nhop (rt (\sigma i)) dip\rangle)))
\land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (\langle nhop (rt (\sigma i)) dip\rangle))) dip"\)
and qinc: "\(\forall j. quality_increases (\sigma j) (\sigma' j)\)"
and *: "sip \(\neq oip\) \rightarrow oip \in kD (rt (\sigma sip))
\land osn \leq nsqn (rt (\sigma sip)) oip
\rightarrow the (dhops (rt (\sigma sip)) oip) \leq hops
\lor the (flag (rt (\sigma sip)) oip) = inv)"

from pre and qinc
have pre': "\(\forall dip \in kD (rt (\sigma i)). \) the \(\langle nhop (rt (\sigma i)) dip\rangle \neq dip\)
\rightarrow dip \in kD (rt (\sigma' (\langle nhop (rt (\sigma i)) dip\rangle)))
\land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma' (\langle nhop (rt (\sigma i)) dip\rangle))) dip"

by (rule basic)
have "\(\forall dip \in kD (rt (\sigma i))\).
the \(\langle nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip\rangle \neq dip\)
\rightarrow dip \in kD (rt (\sigma' (\langle nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {}) dip\rangle))))
\land nsqn (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {})) dip
\leq nsqn (rt (\sigma' (\langle nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip, {}) dip\rangle)))) dip" (is "\(\forall dip \in kD (rt (\sigma i)). \) ...
\rightarrow dip_in_kD dip \land \?nsqn_le_nsqn 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 \land \?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 \land \?nsqn_le_nsqn dip" proof (cases "dip = oip")
assume "dip = 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

} note update2 = this

have "opaodv i \models (\?S, ?U \rightarrow) oln \Gamma_{AODV} (\lambda(\sigma, _). \forall dip \in kD (rt (\sigma i)). the (nhop (rt (\sigma i)) dip) \neq dip
\rightarrow dip \in kD (rt (\sigma (\langle nhop (rt (\sigma i)) dip\rangle)))
\land nsqn (rt (\sigma i)) dip \leq nsqn (rt (\sigma (\langle nhop (rt (\sigma i)) dip\rangle))) dip)"

by (inv_cterms inv add: oseq_step_invariant_sterms [OF quality_increases_nsqn_fresh aodv_wf oaodv_trans]...
thus \( \text{?thesis} \) unfolding Let_def by auto
qed

Proposition 7.30

lemmas okD_unc_or_atleast_one =
  open_seq_invariant [OF kD_unc_or_atleast_one initiali_aodv, simplified seql_onl_swap]

lemmas ozero_seq_unk_hops_one =
  open_seq_invariant [OF zero_seq_unk_hops_one initiali_aodv, simplified seql_onl_swap]

lemma reachable_fresh_okD_unc_or_atleast_one:
  fixes dip
  assumes "\((\sigma, p) \in \text{reachable (opaodv i)}\)
   (\text{otherwith (op=} (i) (orecvmsg (\lambda \sigma m. msg\_fresh \sigma m \\
   \land msg\_zhops m)))
   (\text{other quality\_increases \{i\}})"
   and "dip\in\text{kD}(rt (\sigma i))"
  shows "\(\pi_3(\text{the (rt (\sigma i) dip)}) = \text{unk} \lor 1 \leq \pi_2(\text{the (rt (\sigma i) dip)})\)"
  (is "?P dip")
proof -
  have "\(\exists l. l \in \text{labels } \Gamma_{AODV} p\)" by (metis aodv_ex_label)
  with assms(1) have "\(\forall dip \in \text{kD}(rt (\sigma i)). ?P dip\)"
    by - (drule oinvariant_weakenD [OF okD_unc_or_atleast_one [OF oaodv_trans aodv_trans]],
      auto dest!: otherwith_actionD onlD simp: seqlsimp)
  with ⟨dip\in\text{kD}(rt (\sigma i))⟩ show ?thesis by simp
qed

lemma reachable_fresh_ozero_seq_unk_hops_one:
  fixes dip
  assumes "\((\sigma, p) \in \text{reachable (opaodv i)}\)
   (\text{otherwith (op=} (i) (orecvmsg (\lambda \sigma m. msg\_fresh \sigma m \\
   \land msg\_zhops m)))
   (\text{other quality\_increases \{i\}})"
   and "dip\in\text{kD}(rt (\sigma i))"
  shows "\(\text{sqn (rt (\sigma i) dip) = 0 \longrightarrow sqnf (rt (\sigma i) dip) = unk}
   \land \text{the (dhops (rt (\sigma i) dip)} = 1
   \land \text{the (nhop (rt (\sigma i) dip)} = dip"
  (is "?P dip")
proof -
  have "\(\exists l. l \in \text{labels } \Gamma_{AODV} p\)" by (metis aodv_ex_label)
  with assms(1) have "\(\forall dip \in \text{kD}(rt (\sigma i)). ?P dip\)"
    by - (drule oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaodv_trans aodv_trans]],
      auto dest!: otherwith_actionD onlD simp: seqlsimp)
  with ⟨dip\in\text{kD}(rt (\sigma i))⟩ show ?thesis by simp
qed

lemma seq_nhop_quality_increases':
  shows "\(\text{opaodv i} \models (\text{otherwith (op=} (i)\)}"
proof -

have weaken: 
"∀ P I Q R. P ⊨ (otherwith quality_increases I (orecmsg Q), other quality_increases I → P) 
⇒ P ⊨ (otherwith (op=) I (orecmsg (λσ m. Q σ m ∧ R σ m)), other quality_increases I → P)"

by auto

qed

} note basic = this

{ fix 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(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 ≠ i → σ j = σ' j" by auto

show "rt (σ i) □_dip rt (σ' (the (nhop (rt (σ i)) dip)))" by auto

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 (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

hence False by simp

thus ?thesis ..

next

assume "(the (nhop (rt (σ i)) dip)) ≠ i"

with ⟨vD (rt (σ i))⟩ have "dip ∈ vD (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

}
(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, {})

\( \sqsubseteq \text{dip} \) 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) \( \sqsubseteq \text{dip} \) rt (σ' (the (nhop (rt (σ i)) dip)))"

using a1 ow by - (erule(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 σ' nhop

assume pre: "∀ dip. dip∈vD(rt (σ i)) ∧ dip∈vD(rt (σ (nhop dip))) ∧ nhop dip ≠ dip
dip → rt (σ i) \( \sqsubseteq \text{dip} \) 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) \( \sqsubseteq \text{dip} \) 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) \( \sqsubseteq \text{dip} \) 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) \( \sqsubseteq \text{dip} \) rt (σ' (nhop dip))"

by (metis rt_strictly_fresher_irefl)

qed

} note invalidate = this

{ fix σ 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) \( \sqsubseteq \text{dip} \) 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
 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)))"  
(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 (update (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 simp
next
— alternative to using sip_not_ip
assume [simp]: "sip = i"
show ?thesis
proof
next
assume "sqn (rt (σ i)) oip < osn"
also from \osn ≤ nsqn (rt (σ i)) 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 \( \text{osn} \leq \text{nsqn} (\text{rt} (\sigma \text{ sip})) \text{ oip} \) have "\( \text{osn} < \text{nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ oip} \)"
by simp
thus \(?\text{thesis} .. \) .
qed
thus \(?\text{thesis} \) by (rule rt_strictly_fresher_ltI)

next
assume \( \text{osneq} : \text{"osn = nsqn} (\text{rt} (\sigma' \text{ sip})) \text{ oip} \∧ \text{the} (\text{dhops} (\text{rt} (\sigma' \text{ sip})) \text{ oip}) \leq \text{hops}\)"

have "\( \text{oip} \in \text{kD}\text{(?rt1)} \)" by simp
moreover have "\( \text{nsqn} \text{ ?rt1} \text{ oip} = \text{nsqn} \text{ (?rt2} \text{ oip}) \text{ oip} \)"
proof -
from \( \text{osneq} \) have "\( \text{osn} = \text{nsqn} \text{ (?rt1} \text{ oip}) \) ..
also have "\( \text{osn} = \text{nsqn} \text{ ?rt1} \text{ oip} \)" by (simp add: change)
finally show \(?\text{thesis} \) .
qed

ultimately have "\( \text{?rt1} \subseteq \text{oip} \text{ ?rt2 oip} \)"
by (rule rt_strictly_fresher_eqI)
with \( \langle \text{dip} = \text{oip} \rangle \) show \(?\text{thesis} \) by simp
qed

} note \( \text{rreq_rrep_update = this} \)

have "\( \text{opaodv i} \models \text{(otherwith (op=) \{i\} (orecvmsg} (\lambda \sigma \text{ m. msg_fresh} \sigma \text{ m} \∧ \text{msg_zhops} \text{ m}))}, \text{\ otherquality_increases \{i\} \rightarrow}) \)\n\( \text{onl} \ \text{Γ}_{\text{AODV}} \) (\( \lambda(\sigma, _) \). \forall \text{dip. 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}))))\)
\[ \land \text{the (nhop (rt (σ i)) dip) \neq dip} \]
\[ \rightarrow \text{rt (σ i) \sqsubseteq dip rt (σ (the (nhop (rt (σ i)) dip)))} \]

**proof** (inv_cterms inv add: onl_oinvariant_sterms [OF aodv_wf rreq_sip [THEN weaken]])

- onl_oinvariant_sterms [OF aodv_wf rrep_sip [THEN weaken]]
- onl_oinvariant_sterms [OF aodv_wf rerr_sip [THEN weaken]]
- onl_oinvariant_sterms [OF aodv_wf cosn_rreq [THEN weaken]]
- onl_oinvariant_sterms [OF aodv_wf odsn_rrep [THEN weaken]]
- onl_oinvariant_sterms [OF aodv_wf oaddpreRT_welldefined]

solve: basic_update_0_unk invalidate rreq_rrep_update

simp add: seqlsimp)

fix \(σ, σ' \) p l

assume or: "(σ, p) \in \text{oreachable (opaodv i) (?S i) (other quality_increases \{i\})}
and "other quality_increases \{i\} σ σ'"
and ll: "l \in \text{labels } Γ \text{AODV} p"
and pre: "∀ dip. dip \in vD (rt (σ i)) \land dip \in vD (rt (σ (the (nhop (rt (σ i)) dip))))
\land \text{the (nhop (rt (σ i)) dip) \neq dip}
\rightarrow \text{rt (σ i) \sqsubseteq dip rt (σ (the (nhop (rt (σ i)) dip))))}

from this(1-2)

have or': "(σ', p) \in \text{oreachable (opaodv 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 \in kD (rt (σ i)) \land nhip \neq dip
\rightarrow dip \in kD (rt (σ nhip))
\land \text{nsqn (rt (σ i)) dip} ≤ \text{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 \in kD (rt (σ i)). sqn (rt (σ i)) dip = 0
\rightarrow sqnf (rt (σ i)) dip = unk
\land \text{the (dhops (rt (σ i)) dip) = 1}
\land \text{the (nhop (rt (σ i)) dip) = dip}"
by (auto dest!: onl_oinvariant_weakenD [OF ozero_seq_unk_hops_one [OF oaodv_trans aodv_trans]]
otherwith_actionD
simp: seqlsimp)

have "∀ j. quality_increases (σ j) (σ' j)
by (erule otherE, metis singleton_iff)

show "∀ dip. dip \in vD (rt (σ' i))
\land dip \in vD (rt (σ' (the (nhop (rt (σ' i)) dip))))
\land \text{the (nhop (rt (σ' i)) dip) \neq dip}
\rightarrow rt (σ' i) \sqsubseteq dip rt (σ' (the (nhop (rt (σ' i)) dip))))"
proof clarify

fix dip

assume "dip \in vD (rt (σ' i))"
and "dip \in vD (rt (σ' (the (nhop (rt (σ' i)) dip))))"
and "the (nhop (rt (σ' i)) dip) \neq dip"
from this(1) and (σ' i = σ i) have "dip \in vD (rt (σ i))"
and "dip \in kD (rt (σ i))"
by auto

from (the (nhop (rt (σ' i)) dip) \neq dip) and (σ' i = σ i)
have "the (nhop (rt (σ i)) dip) \neq dip" (is "?nhip \neq _") by simp
with "dip \in kD (rt (σ i))" and next_hop
have "dip \in kD (rt (σ (?nhip)))"
and nsqns: "\text{nsqn (rt (σ i)) dip} ≤ \text{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])

405
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 ⟨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∈vD(rt (σ i))⟩ and ⟨?nhip ≠ dip⟩
    have "rt (σ i) ⊏ dip rt (σ ?nhip)" by auto
  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 vB_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)
    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} (orecmsg 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_nhopt_quality_increases:
shows "opaodv 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_nhopt_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 σ, 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, 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 = (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 nhip))" and "nhip = the (nhop (rt (σ ip)) dip)" by auto
from ⟨dip ∈ vD(rt (σ nhip))⟩ and ⟨dip ∈ vD(rt (σ nhip'))⟩ have "dip ∈ vD(rt (σ nhip)) ∩ vD(rt (σ nhip'))" ..
with nhip = the (nhop (rt (σ 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)⁺" and "ip ≠ dip"
by auto
ultimately have "rt (σ ip) ⊑ dip rt (σ ip)" by (rule fresher)
thus False by simp

qed
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 σ s a σ' s'
  assumes "((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos Γ_AODV i) (seqp_sos Γ_QMSG)"
  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 ⟩⟩ = A (otherwith (op=) {i} (orecvmsg (λs. msg_fresh s ∧ 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 ⟨⟨ i qmsg ⟩⟩ = A (otherwith (op=) {i} (orecvmsg (λs. msg_fresh s ∧ msg_zhops m)), other quality_increases {i} →)
    globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
    proof (rule ostep_invariant_weakenE [OF olocal_quality_increases], simp_all)
      fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
      assume "onll Γ_AODV (λ((σ, _, _), _, (σ', _)). 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 (λs. msg_fresh s ∧ 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

lemma par_rreq_rrep_sn_quality_increases:
  "opaodv i ⟨⟨ i qmsg ⟩⟩ = A (λs _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →)
  globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
  proof -
    have "opaodv i ⟨⟨ i qmsg ⟩⟩ = A (λs _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →)
    globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
      by (rule ostepInvariant_weakenE [OF olocal_quality_increases] simp_all)
    hence "opaodv i ⟨⟨ i qmsg ⟩⟩ = A (λs _. 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 "opaodv i ⟨⟨ i qmsg ⟩⟩ = A (λs _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →)
  globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)"
  proof -
have \( \text{opaodv} \models_A (\lambda \sigma \_ . \text{orecvmsg} (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma) \), other \( (\lambda \_ \_ . \text{True}) \) \( \{i\} \rightarrow \)
globala \( (\lambda (\sigma, a, \_'). \text{anycast} (\lambda \_ . \_ . \text{msg\_fresh} \sigma) a) \)

proof (rule ostep\_invariant\_weakenE \[OF rreq\_rrep\_msq\_fresh\_any\_step\_invariant\])

fix \( t \)
assume \( \text{onll} \Gamma_{AODV} (\lambda((\sigma, \_), a, \_). \text{anycast} (\lambda \_ . \_ . \text{msg\_fresh} \sigma) a) t \)
thus \( \text{globala} (\lambda (\sigma, a, \_'). \text{anycast} (\lambda \_ . \_ . \text{msg\_fresh} \sigma) a) t \)
by (cases \( t \)) (clarsimp dest!: onllD, metis aodv_ex_label)
qed auto

hence \( \text{opaodv} \models_A (\lambda \sigma \_ . \text{orecvmsg} (\lambda \_ . \text{rreq\_rrep\_sn}) \sigma) \), other \( (\lambda \_ \_ . \text{True}) \) \( \{i\} \rightarrow \)
globala \( (\lambda (\sigma, a, \_'). \text{anycast} (\lambda \_ . \_ . \text{msg\_fresh} \sigma) a) \)" by (rule lift\_step\_into\_qmsg\_statelessassm) simp_all
thus \( \text{?thesis} \) by rule auto
qed

4.11.2 Lift to nodes

lemma \( \text{node\_step\_no\_change\_on\_send\_or\_receive} \):
assumes \( \langle \langle \sigma, \text{NodeS} i P R \rangle, a, (\sigma', \text{NodeS} i' P' R') \rangle \in \text{onode\_sos} (\text{oparp\_sos} i (\text{oseqp\_sos} \Gamma_{AODV} i) (\text{seqp\_sos} \Gamma_{QM SG})) \) and \( a \neq \tau \)
shows \( \sigma' \_ i = \sigma \_ i \)
using \( \text{assms} \)
by (cases \( a \)) (auto elim!: par\_step\_no\_change\_on\_send\_or\_receive)

lemma \( \text{node\_nhop\_quality\_increases} \):
shows \( \langle \langle \sigma, \text{NodeS} i (\_ \_ P R) \rangle, a, (\sigma', \text{NodeS} i' (\_ \_ P' R')) \rangle \in \text{onode\_sos} (\text{oparp\_sos} \_ i (\text{oseqp\_sos} \Gamma_{AODV} i) \_ \_ i) \_ \_ \_ \_ i) \)
by (rule node\_lift \[OF \text{par\_nhop\_quality\_increases} \]) auto

lemma \( \text{node\_quality\_increases} \):
shows \( \langle \langle \sigma, \text{NodeS} i (\_ \_ P R) \rangle, a, (\sigma', \text{NodeS} i' (\_ \_ P' R')) \rangle \in \text{onode\_sos} (\text{oparp\_sos} \_ i (\text{oseqp\_sos} \Gamma_{AODV} i) \_ \_ i) \_ \_ \_ \_ i) \)
by (rule node\_lift\_step\_statelessassm \[OF \text{par\_rreq\_rrep\_sn\_quality\_increases} \]) simp

lemma \( \text{node\_rreq\_rrep\_msq\_fresh\_any\_step} \):
shows \( \langle \langle \sigma, \text{NodeS} i (\_ \_ P R) \rangle, a, (\sigma', \text{NodeS} i' (\_ \_ P' R')) \rangle \in \text{onode\_sos} (\text{oparp\_sos} \_ i (\text{oseqp\_sos} \Gamma_{AODV} i) \_ \_ i) \_ \_ \_ \_ i) \)

(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 : opaodv i |{i qmsg : R} >>= )
(λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →)
globala (λ(σ, a, σ'). castmsg msg_zhops a)"
by (rule node_lift_anycast_statelessassm [OF par_anycast_msg_zhops])

**lemma node_silent_change_only:**
shows "(( i : opaodv i |{i qmsg : R} >>= )
(λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →)
globala (λ(σ, a, σ'). a ≠ τ → σ' i = σ i)"
proof (rule ostep_invariantI, simp (no_asm), rule impl)
fix σ a σ' a'
assume or: "((σ, ζ) ∈ reachable (( i : opaodv i |{i qmsg : R} >>= )
(λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ)
(other (λ_. True) {i}))
and tr: "((σ, ζ), a, (σ', ζ')) ∈ trans (( i : opaodv i |{i qmsg : R} >>= )"
and "a ≠ τ n"
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 (opaodv i))) (trans qmsg))"
  by simp
thus "σ' i = σ i" using "a ≠ τ n" 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_nhloop_quality_increases:**
shows "opnet (λi. opadov i |{i qmsg} >>=
(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 node_lift [OF node_nhloop_quality_increases])
fix i R
have "(( i : opaodv i |{i qmsg : R} >>= )
(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 a σ' a'
assume or: "((σ, s) ∈ reachable (( i : opaodv i |{i qmsg : R} >>= )
(λσ _. oarrivemsg (λ_. rreq_rrep_sn) σ)
(other (λ_. True) {i}))
and tr: "((σ, s), a, (σ', s')) ∈ trans (( i : opaodv i |{i qmsg : R} >>= )"
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_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 "\langle i : \text{opaodv} \ i \rangle \models \lambda \sigma \_ . \ \text{oarrivemsg} \ (\lambda \sigma \ m . \ \text{msg\_fresh} \ \sigma \ m \ \land \ \text{msg\_zhops} \ m) \ \sigma , \ \text{other\ quality\_increases} \ \{ i \} \rightarrow) \ \text{globala} \ (\lambda (\sigma, \ a, \ _)) . \ \text{c castmsg} \ (\lambda m . \ \text{msg\_fresh} \ \sigma \ m \ \land \ \text{msg\_zhops} \ m) \ a)"

by rule auto

next

fix \ i \ R

show "\langle i : \text{opaodv} \ i \rangle \models \lambda \sigma \_ . \ \text{oarrivemsg} \ (\lambda \sigma \ m . \ \text{msg\_fresh} \ \sigma \ m \ \land \ \text{msg\_zhops} \ m) \ \sigma , \ \text{other\ quality\_increases} \ \{ i \} \rightarrow) \ \text{globala} \ (\lambda (\sigma, \ a, \ \sigma')) . \ \sigma = \tau \ \land \ \forall i \ d . \ a \neq i : \text{deliver}(d) \ \rightarrow \ \sigma \ i \neq \sigma' \ i)"

by (rule ostep\_invariant\_weakenE [OF node\_silent\_change\_only]) auto

next

fix \ i \ R

show "\langle i : \text{opaodv} \ i \rangle \models \lambda \sigma \_ . \ \text{oarrivemsg} \ (\lambda \sigma \ m . \ \text{msg\_fresh} \ \sigma \ m \ \land \ \text{msg\_zhops} \ m) \ \sigma , \ \text{other\ quality\_increases} \ \{ i \} \rightarrow) \ \text{globala} \ (\lambda (\sigma, \ a, \ \sigma') . \ \text{a} = \tau \ \lor \ \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

4.11.4 Lift to closed networks

lemma onet\_nhop\_quality\_increases:

shows "\text{oclosed} \ (\text{opnet} \ (\lambda i . \ \text{opaodv} \ i \langle i \ qmsg \rangle) \ p) \models \ (\lambda \_ \ _ . \ \text{True}, \ \text{other\ quality\_increases} \ (\text{net\_tree\_ips} \ p) \rightarrow) \ \text{globala} \ (\lambda \sigma . \ \forall i \in \text{net\_tree\_ips} \ p . \ \forall 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)) \ \land \ \text{vD} \ (\text{rt} \ (\sigma \ nhip)) \ \land \ nhip \neq dip \ \rightarrow \ (\text{rt} \ (\sigma \ i)) \ \subseteq \text{dip} \ (\text{rt} \ (\sigma \ nhip)))"

(is "\_ \models \ (\_ , \ ?U \rightarrow) \ ?inv")

proof (rule inclosed\_closed)

from opnet\_nhop\_quality\_increases

show "\text{opnet} \ (\lambda i . \ \text{opaodv} \ i \langle i \ qmsg \rangle) \ p \models \ (\text{otherwith} \ (\text{op=} \) \ (\text{net\_tree\_ips} \ p) \ \text{inoclosed}, \ ?U \rightarrow) \ ?inv"

proof (rule oinvariant\_weakenE)

fix \ \sigma \ \sigma' :: \ "\text{ip} \ \Rightarrow \ \text{state}" \ and \ \text{a} :: \ "\text{msg} \ \text{node\_action}"

assume "\text{otherwith} \ (\text{op=} \) \ (\text{net\_tree\_ips} \ p) \ \text{inoclosed} \ \sigma \ \sigma' \ \text{a}"

thus "\text{otherwith} \ (\text{op=} \) \ (\text{net\_tree\_ips} \ p) \ \\text{(oarrivemsg} \ (\lambda \sigma \ m . \ \text{msg\_fresh} \ \sigma \ m \ \land \ \text{msg\_zhops} \ m) \ \sigma , \ \text{other\ quality\_increases} \ \{ i \} \rightarrow) \ \text{globala} \ (\lambda (\sigma, \ a, \ \sigma') . \ \text{a} = \tau \ \lor \ \exists d . \ a = i : \text{deliver}(d) \ \rightarrow \ \text{quality\_increases} \ (\sigma \ i) \ (\sigma' \ i))"

proof (rule otherwith\_EI)

fix \ \sigma :: \ "\text{ip} \ \Rightarrow \ \text{state}" \ and \ \text{a} :: \ "\text{msg} \ \text{node\_action}"

assume "\text{inoclosed} \ \sigma \ \text{a}"

thus "\text{oarrivemsg} \ (\lambda \sigma \ m . \ \text{msg\_fresh} \ \sigma \ m \ \land \ \text{msg\_zhops} \ m) \ \sigma \ \text{a}"

proof (cases \ \text{a})

fix \ \text{ii} \ \text{ni} \ \text{ms}

assume "\text{a} = \text{ii}::\text{ni}.\text{arrive}(	ext{ms})"

moreover with \text{inoclosed} \ \sigma \ \text{a} \ obtain \ d \ \text{dip} \ where \ "\text{ms} = \text{newpkt}(\text{d}, \ \text{di})"

by (cases \ \text{ms}) auto

ultimately show ?thesis by simp

qed simp\_all

qed

qed

4.11.5 Transfer into the standard model

interpretation aodv\_openproc: openproc \ paodv \ opaodv \ id

where "aodv\_openproc.initmissing = initmissing"

proof -

show "openproc \ paodv \ opaodv \ id"

proof unfold\_locales
fix i :: ip

have "\{ (σ, ζ). (σ i, ζ) ∈ σ_{AODV} AODV i ∧ (∀ j ≠ i → σ j ∈ fst ' σ_{AODV} j) \} ⊆ σ_{AODV}" 
  unfolding σ_{AODV_def} σ_{AODV'_def} 
proof (rule equalityD1) 
  show "\{ f p. (σ, ζ). (σ i, ζ) ∈ {f i, p} ∧ (∀ j ≠ i → σ j ∈ fst ' {f j, p}) \} = \{f, p\}" 
    by (rule set_eqI) auto 
qed 

thus "\{ (σ, ζ) | σ ζ s. s ∈ init (paodv i) ∧ (∀ j ≠ i → σ j ∈ init (paodv j)) \} ⊆ init (opaodv i)" 
  by simp 

next 
show "∀ j. init (paodv 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 (paodv i)" 
then obtain q q' where "s = (σ i, q)" 
  and "s' = (σ' i, q')" 
  and "((σ i, q), a, (σ' i, q')) ∈ trans (paodv i)" 
    by (cases s, cases s') auto 
from this(3) have "((σ, q), a, (σ', q')) ∈ trans (opaodv i)" 
    by (rule open_seqp_action [OF aodv_wf]) 
with ⟨s = (σ i, q)⟩ and ⟨s' = (σ' i, q')⟩ 
show "((σ, snd (id s)), a, (σ', snd (id s'))) ∈ trans (opaodv i)" 
  by simp 
qed 

interpretation adv: openproc_parq aodv opaodv id qmsg 
where "adv.netglobal = netglobal" 
  and "adv.initmissing = initmissing" 
proof - 
show "openproc_parq aodv opaodv id qmsg" 
  by (unfold_locales) simp 
then interpret opq: openproc_parq aodv opaodv id qmsg . 

have im: "\{ σ. openproc.initmissing (λ i. paodv i (λ (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 (λ i. paodv i (λ (qmsg) (λ (p, q). (fst (id p), snd (id p), q)))) = initmissing" 
  by (rule ext) 
have "\{ P σ. openproc.netglobal (λ i. paodv i (λ (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 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_netmap_default_aodv_init_netlift [symmetric, unfolded initmissing_def]) 
thus "openproc.netglobal (λ i. paodv i (λ (qmsg) (λ (p, q). (fst (id p), snd (id p), q)))) = netglobal" 
  by auto 

413
lemma net_nhop_quality_increases:
assumes "wf_net_tree n"
shows "closed (pnet (λi. paodv 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. paodv 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 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 σ i
    assume sr: "σ ∈ reachable (closed (pnet (λi. paodv i (( qmsg) n))) TT"
    hence "∀ i∈net_tree_ips n. ?inv (fst (initmissing (netgmap fst σ))) i"
      by - (drule invariantD [OF proto],
        simp only: aodv_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. paodv 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 = 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 (λi. paodv i (( qmsg) n)) |= netglobal (λσ. ∀ dip. irrefl ((rt_graph σ dip)⁺))"
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 ⇒ sqn"
  where "inc sn ≡ if sn = 0 then sn else sn + 1"

lemma less_than_inc [simp]: "x ≤ 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 ≠ Suc 0"
  unfolding inc_def by simp

lemma inc_never_one [simp, intro]: "inc x ≠ 1"
  by simp

5.1.2 Modelling Routes

A route is a t-tuple, (dsn, dsk, flag, hops, nhip) 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 nhip is the next hop toward the destination.

type_synonym r = "sqn × k × f × nat × 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 \( \pi_6 \equiv \lambda(_, _, _, _, nhip). nhip \)

lemma projs [simp]:
\[
\begin{align*}
\pi_2&(dsn, dsk, flag, hops, nhip) = dsn \\
\pi_3&(dsn, dsk, flag, hops, nhip) = dsk \\
\pi_4&(dsn, dsk, flag, hops, nhip) = flag \\
\pi_5&(dsn, dsk, flag, hops, nhip) = hops \\
\pi_6&(dsn, dsk, flag, hops, nhip) = nhip
\end{align*}
\]

by (clarsimp simp: proj2_def proj3_def proj4_def proj5_def proj6_def)+

lemma proj3_pred [intro]: "[ \[ P kno; P unk \] ] \implies P (\pi_3 x)"
by (rule k.induct)

lemma proj4_pred [intro]: "[ \[ P val; P inv \] ] \implies P (\pi_4 x)"
by (rule f.induct)

lemma proj6_pair_snd [simp]:
fixes dsn' r
shows \( \pi_6 (dsn', \operatorname{snd}(r)) = \pi_6 (r) \)
by (cases r) simp

5.1.3 Routing Tables
Routing tables map ip addresses to route entries.

type synonym \( rt = \text{"ip \rightarrow r"} \)

syntax
\[ \_\Sigma\text{route} :: rt \Rightarrow ip \leftrightarrow r \text{"(\"route'(_, _')\"')}\]

translations
\[ \sigma\text{route}(rt, dip) \Rightarrow rt dip \]

definition sqn :: "rt \Rightarrow ip \Rightarrow sqn"
where "sqn rt dip \equiv \operatorname{case} \sigma\text{route}(rt, dip) of \text{Some \(r\) \Rightarrow \pi_2 (r) | \text{None \Rightarrow 0}}"

definition sqnf :: "rt \Rightarrow ip \Rightarrow k"
where "sqnf rt dip \equiv \operatorname{case} \sigma\text{route}(rt, dip) of \text{Some \(r\) \Rightarrow \pi_3 (r) | \text{None \Rightarrow unk}}"

abbreviation flag :: "rt \Rightarrow ip \leftrightarrow f"
where "flag rt dip \equiv \operatorname{map\_option} \pi_4 (\sigma\text{route}(rt, dip))"

abbreviation dhops :: "rt \Rightarrow ip \leftrightarrow nat"
where "dhops rt dip \equiv \operatorname{map\_option} \pi_5 (\sigma\text{route}(rt, dip))"

abbreviation nhop :: "rt \Rightarrow ip \leftrightarrow ip"
where "nhop rt dip \equiv \operatorname{map\_option} \pi_6 (\sigma\text{route}(rt, dip))"

definition vD :: "rt \Rightarrow ip set"
where "vD rt \equiv \{dip. flag rt dip = \text{Some \(val\)}\}"

definition iD :: "rt \Rightarrow ip set"
where "iD rt \equiv \{dip. flag rt dip = \text{Some \(inv\)}\}"

definition kD :: "rt \Rightarrow ip set"
where "kD rt \equiv \{dip. rt dip \neq \text{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]:
\[
\begin{align*}
\forall ip \ rt. \ ip \in vD \ rt \implies ip \in kD \ rt \\
\forall ip \ rt. \ ip \in iD \ rt \implies ip \in kD \ rt
\end{align*}
\]
unfolding kD_is_vD_and_iD by simp_all

416
lemma kD_Some [dest]:
  fixes dip rt
assumes "dip \in kD rt"
sshows "\exists dsn dsk flag hops nhip.
    \sigma_{route}(rt, dip) = Some (dsn, dsk, flag, hops, nhip)"
using assms unfolding kD_def by simp

lemma kD_None [dest]:
  fixes dip rt
assumes "dip \notin kD rt"
sshows "\sigma_{route}(rt, dip) = 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"
sshows "\exists dsn dsk hops nhip.
    \sigma_{route}(rt, dip) = Some (dsn, dsk, val, hops, nhip)"
using assms unfolding vD_def by simp

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

lemma iD_Some [dest]:
  fixes dip rt
assumes "dip \in iD rt"
sshows "\exists dsn dsk hops nhip.
    \sigma_{route}(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 \in kD(rt)"
and "the (flag rt ip) = val"
sshows "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"
sshows "ip \in iD(rt)"
using assms unfolding iD_def by auto

lemma iD_flag_is_inv [elim, simp]:
  fixes ip rt
assumes "ip \in iD(rt)"
sshows "the (flag rt ip) = inv"
proof -
  from ⟨ip \in iD(rt)⟩ have "ip \in 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 \in kD(rt)"
and "ip \notin vD(rt)"
sshows "ip \in iD(rt)"
proof -
  from ⟨ip \in kD(rt)⟩ obtain dsn dsk f hops nhop
  where rtip: "rt ip = Some (dsn, dsk, f, hops, nhop)"
  by (metis kD_Some)
from \(\{\text{ip} \notin vD(rt)\}\) have "f ≠ val"

proof (rule contrapos_nn)
  assume "f = val"
  with rtip have "the (flag rt ip) = val" by simp
  with \(\text{ip} \in kD(rt)\) show "ip \in vD(rt)" ..
qed

with rtip have "the (flag rt ip) = inv" by simp
with \(\text{ip} \in kD(rt)\) show "ip \in iD(rt)" ..
qed

lemma vD_or_iD [elim]:
  fixes ip rt
  assumes "\text{ip} \in kD(rt)"
  and "\text{ip} \in vD(rt) \Rightarrow P \text{ rt ip}"
  and "\text{ip} \in iD(rt) \Rightarrow P \text{ rt ip}"
  shows "P \text{ rt ip}"
proof (-
  from \(\text{ip} \in kD(rt)\) have "\text{ip} \in vD(rt) \cup iD(rt)"
    by (simp add: kD_is_vD_and_iD)
  thus ?thesis by (auto elim: assms(2-3))
qed

5.1.4 Updating Routing Tables

Routing table entries are modified through explicit functions. The properties of these functions are important in invariant proofs.
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)"
"∀dsn 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 −→ (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 \(\forall\)thesis using \(\forall\flag = val\) by simp (rule c1)
next
assume "dsk = kno"
moreover with wf2 have "dsn > 0" by simp
ultimately show \(\forall\)thesis using \(\forall\flag = val\) by simp (rule c2)
qed
with * show "P r" by simp
definition update :: "rt ⇒ ip ⇒ r ⇒ rt"

where

"update rt ip r ≡
  case σ_route(rt, ip) of
    None ⇒ rt (ip ↦→ r)
  | Some s ⇒
    if π₂(s) < π₂(r) then rt (ip ↦→ r)
    else if π₂(s) = π₂(r) ∧ (π₅(s) > π₅(r) ∨ π₄(s) = inv)
      then rt (ip ↦→ r)
    else if π₃(r) = unk
      then rt (ip ↦→ (π₃(s), snd (r)))
    else rt (ip ↦→ s)"

lemma update_simps [simp]:
  fixes r s nrt nr' ns rt ip
  defines "s ≡ the σ_route(rt, ip)"
  and "nr' ≡ (π₂(s), π₃(r), π₄(r), π₅(r), π₆(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 ↦→ r)"
    "[ [ ip ∈ kD(rt); sqn rt ip = π₂(r); the (dhops rt ip) > π₅(r) ] ] =⇒ update rt ip r = rt (ip ↦→ r)"
    "[ [ 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 ↦→ s)"

proof -
  assume "ip /∈ kD(rt)"
  hence "σ_route(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
    where "rt ip = Some (dsn, dsk, fl, hops, nhip)"
    by (metis kD_Some)
  with ⟨sqn rt ip < π₂(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 "the (dhops rt ip) > π₅(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 = π₂(r)⟩ and ⟨the (dhops rt ip) > π₅(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 ↦ (π₃(r) =⇒ the (dhops rt ip) ≤ π₅(r) ∧ the (flag rt ip) = val )
  =⇒ P (rt (ip ↦ the σ route (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 ↦ the σ route (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 rreq: "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) = unk \implies P (rt (ip \mapsto (\pi_2(\sigma_{route}(rt, ip)), \pi_3(r), 
\pi_4(r), \pi_5(r), \pi_6(r))))"
and c6: "[sqn rt ip $\geq$ $\pi_2(r)$; $\pi_3(r) = kno$; 
sqn rt ip = $\pi_2(r)$ $\implies$ the (dhops rt ip) $\leq$ $\pi_5(r)$ 
and the (flag rt ip) = val] 
\implies P (rt (ip \mapsto the (\sigma_{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 "$\pi_3(r) = unk$"
thus "P (rt (ip \mapsto (\pi_2(\sigma_{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 "$\pi_3(r) = kno$"
and "sqn rt ip = $\pi_2(r)$ $\implies$ the (dhops rt ip) $\leq$ $\pi_5(r)$ 
and the (flag rt ip) = val"
thus "P (rt (ip \mapsto 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: "$\\forall v. rt dip = Some v \implies 
update rt dip (dsn, dsk, flag, hops, nhip) 
\neq rt(dip \mapsto 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 (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 flag hops nhip
  assumes "(dsn = 0) = (dsk = unk)"
  shows "sqn rt dip ≤ sqn (update rt rip (dsn, dsk, val, hops, nhip)) dip"
proof (rule update_cases)
  show "(π2 (dsn, dsk, val, hops, nhip) = 0) = (π3 (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 ≤ hops"
  shows "sqn rt ip ≤ 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: "∀ 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)) 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
  assumes "ip ≠ 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 ≠ 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 flg hops sip.
  P (update rt ip (dsn, dsk, flg, hops, sip)) =
  (¬(rt = update rt ip (dsn, dsk, flg, hops, sip) ∧ ¬P rt
  ∨ rt ≠ update rt ip (dsn, dsk, flg, hops, sip)
  ∧ ¬P (update rt ip (dsn, dsk, flg, 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 \in kD(rt)"
  proof -
    have "dip \in kD(update rt dip (dsn, dsk, flag, hops, nhip))" by simp
    with assms show thesis by simp
  qed

lemma nhop_update [simp]: "\forall 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 = sip"
unfolding update_def sqnf_def
by (clarsimp split: option.split_if_asm) auto

lemma sqn_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip \neq 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.split) auto

lemma sqnf_update_another [simp]:
  fixes dip ip rt dsn dsk flag hops nhip
  assumes "ip \neq 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.split) auto

lemma vD_update_val [dest]:
  "\forall dip rt dip' dsn dsk hops nhip. dip \in vD(update rt dip' (dsn, dsk, val, hops, nhip)) \implies (dip \in vD(rt) \lor dip=dip')"
unfolding update_def vD_def by (clarsimp split: option.split_asm split_if_asm)

Invalidating route entries

definition invalidate :: "rt \Rightarrow (ip \mapsto sqn) \Rightarrow rt"
where "invalidate rt dests \equiv \lambda ip. case (rt ip, dests ip) of
  (None, _) \Rightarrow None
  | (Some s, None) \Rightarrow Some s
  | (Some (_, dsk, _, hops, nhip), Some rsn) \Rightarrow Some (rsn, dsk, inv, hops, nhip)"

lemma proj3_invalidate [simp]:
  "\forall dip. \pi_3 (the ((invalidate rt dests) dip)) = \pi_3 (the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj5_invalidate [simp]:
  "\forall dip. \pi_5 (the ((invalidate rt dests) dip)) = \pi_5 (the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)

lemma proj6_invalidate [simp]:
  "\forall dip. \pi_6 (the ((invalidate rt dests) dip)) = \pi_6 (the (rt dip))"
unfolding invalidate_def by (clarsimp split: option.split)

5.1.5 Route Requests

lemma invalidate_kD_inv [simp]:
  "\forall 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 π₁(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)

assms [simp]:
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" ("σ queue’(_, _)’")
where "σ queue (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 := None)
| 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" ("σ p-flag’(_, _)’")
where "σ p-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 ≡ λ dip. 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. 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 (\( \text{dip} \)) 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 dests mapping passed to \textit{invalidate}, and \textit{store}.

### 5.2 AODV protocol messages

```plaintext
theory \textit{E_Aodv_Message} imports \textit{E_All_ABCD} begin
datatype \textit{msg} =
  \textit{Rreq} \text{nat} \text{ip} \text{sqn} k \text{ip} \text{sqn} \text{ip} \text{bool}
  | \textit{Rrep} \text{nat} \text{ip} \text{sqn} \text{ip} \text{ip}
  | \textit{Rerr} \text{"ip \Rightarrow sqn"} \text{ip}
  | \textit{Newpkt} \text{data} \text{ip}
  | \textit{Pkt} \text{data} \text{ip} \text{ip}

instantiation \textit{msg} :: \textit{msg} begin
  definition \textit{newpkt_def} \[\text{simp}\]: \("\text{newpkt} \equiv \lambda(d, dip). \text{Newpkt} d dip"
  definition \textit{eq_newpkt_def}: \"\text{eq\_newpkt} m \equiv \text{case} m \text{ of} \text{Newpkt} d dip \Rightarrow \text{True} | _ \Rightarrow \text{False}"

  instance by intro_classes \(\text{simp add: eq\_newpkt\_def}\)
end
```

The \textit{msg} type models the different messages used within AODV. The instantiation as a \textit{msg} is a technicality due to the special treatment of \textit{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.

```plaintext
definition \textit{rreq} :: \"\text{nat} \times \text{ip} \times \text{sqn} \times k \times \text{ip} \times \text{sqn} \times \text{ip} \times \text{bool} \Rightarrow \text{msg}\"
  where \"\text{rreq} \equiv \lambda(hops, dip, dsn, dsk, oip, osn, sip, handled). \text{Rreq} hops dip dsn dsk oip osn sip handled"

lemma \textit{rreq\_simp} \[\text{simp}\]:
  \"\text{rreq}(hops, dip, dsn, dsk, oip, osn, sip, handled) = \text{Rreq} hops dip dsn dsk oip osn sip handled\"
  unfolding \textit{rreq\_def} by simp

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

lemma \textit{rrep\_simp} \[\text{simp}\]:
  \"\text{rrep}(hops, dip, dsn, oip, sip) = \text{Rrep} hops dip dsn oip sip\"
  unfolding \textit{rrep\_def} by simp

definition \textit{rerr} :: \"(\text{ip} \Rightarrow \text{sqn}) \times \text{ip} \Rightarrow \text{msg}\"
  where \"\text{rerr} \equiv \lambda(dests, sip). \text{Rerr} dests sip"

lemma \textit{rerr\_simp} \[\text{simp}\]:
  \"\text{rerr(dests, sip)} = \text{Rerr dests sip}\"
  unfolding \textit{rerr\_def} by simp

lemma \textit{not\_eq\_newpkt\_rreq} \[\text{simp}\]: \"\neg \text{eq\_newpkt} \ (\text{Rreq} hops dip dsn dsk oip osn sip handled)\"
  unfolding \textit{eq\_newpkt\_def} by simp

lemma \textit{not\_eq\_newpkt\_rrep} \[\text{simp}\]: \"\neg \text{eq\_newpkt} \ (\text{Rrep} hops dip dsn oip sip)\"
  unfolding \textit{eq\_newpkt\_def} by simp
```

428
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)\n  |)“
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),
                   dip := (SOME x. True),
                   oip := (SOME x. True),
                   dsn := (SOME x. True),
                   dsk := (SOME x. True),
                   osn := (SOME x. True),
                   sip := (SOME x. x ≠ ip ξ),
                   handled:= (SOME x. True))"

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

5.3.2 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' dip' dsn' dsk' oip' osn' sip' handled' ⇒
    { ξ( | hops := hops', 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' dip' dsn' dsk' oip' osn' sip' handled'.
  msg ξ = Rreq hops' dip' dsn' dsk' oip' osn' sip' handled' ∧
  ξ' = ξ( | hops := hops', 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
lemma is_rrep_asm [dest!]:
assumes "ξ' ∈ is_rrep ξ"
shows "∃hops' dip' dsn' oip' sip'.
msg ξ = Rrep hops' dip' dsn' oip' 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_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 =
  PAodv
| PNewPkt
| PPkt
| PRreq
| PRrep
| PRerr

fun nat_of_seqp :: "pseqp ⇒ 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, dip, dsn, dsk, oip, osn, sip, handled) = args ξ in
    (clear_locals ξ) (| hops := hops, dip := dip,
abbreviation RERR
where
"RERR args ≡\\[
  \xi. \text{let } (\text{hops}, \text{dip}, dsn, oip, sip) = \text{args }\xi \text{ in }\\
  (clear_locals \xi) (\text{hops} := \text{hops}, \text{dip} := \text{dip}, dsn := dsn, oip := oip, sip := sip)\\
call(PRrep)"
\]

abbreviation RREP
where
"RREP args ≡\\[
  \xi. \text{let } (\text{dests}, sip) = \text{args }\xi \text{ in }\\
  (clear_locals \xi) (\text{dests} := \text{dests}, sip := sip)\\
call(PRrep)"
\]

fun Γ_{AODV} :: "((state, msg, pseqp, pseqp label) seqp_env)" where
"Γ_{AODV} \text{PAodv} = labelled \text{PAodv} (\\
  \text{receive}(\lambda \text{msg}' . \xi (\text{msg} := \text{msg}'))) .\\
  (\text{is_newpkt}) \text{NEWPKT}(\lambda \xi . (\text{data }\xi, \text{ip }\xi))\\
  + (\text{is_pkt}) \text{PKT}(\lambda \xi . (\text{data }\xi, \text{dip }\xi, oip ))\\
  + (\text{is_rreq})\\
  \quad (rreq) [\\
    \xi. \xi (\text{rt} := \text{update}(\text{rt }\xi) (\text{sip }\xi) (0, \text{unk}, 1, \text{ip }\xi)) ]\\
  \quad \text{RREQ}(\lambda \xi . (\text{hops }\xi, \text{dip }\xi, dsn \xi, dkxi, oip \xi, osn \xi, sip \xi, handled ))\\
  + (\text{is_rrep})\\
  \quad (rrep) [\\
    \xi. \xi (\text{rt} := \text{update}(\text{rt }\xi) (\text{sip }\xi) (0, \text{unk}, 1, \text{ip }\xi)) ]\\
  \quad \text{RREP}(\lambda \xi . (\text{hops }\xi, \text{dip }\xi, dsn \xi, oip \xi, sip \xi))\\
  + (\text{is_rerr})\\
  \quad (rerr) [\\
    \xi. \xi (\text{rt} := \text{update}(\text{rt }\xi) (\text{sip }\xi) (0, \text{unk}, 1, \text{ip }\xi)) ]\\
  \quad \text{RERR}(\lambda \xi . (\text{dests }\xi, sip ) )\\
)\\
+ (\lambda \xi . (\xi (\text{dip} := \text{dip }) ) (\text{dip }\xi \in \text{qD}(\text{store }\xi) \cap \text{vD}(\text{rt }\xi)) )\\
\quad [\\
  \xi . \xi (\text{data} := \text{hd}(\text{\sigma_q\_queue}(\text{store }\xi, \text{dip }\xi)) ) ]\\
\quad \text{unicast}(\lambda \xi . \text{the (nhop (rt }\xi) (\text{dip }\xi)), \lambda \xi . \text{pkt(data }\xi, \text{dip }\xi, \text{ip }\xi)) .\\
\quad [\\
  \xi . \xi (\text{store} := \text{the (drop (dip }\xi) (\text{store }\xi)) ) ]\\
\quad \text{AODV()}\\
+ (\lambda \xi . (\xi (\text{dip} := \text{dip }) ) (\text{dip }\xi \in \text{qD}(\text{store }\xi) \cap \text{vD}(\text{rt }\xi) \cap \text{the (\sigma_p\_flag (\text{store }\xi, \text{dip }) ) = \text{req } }) )\\
\quad [\\
  \xi . \xi (\text{store} := \text{unsetRRF (store }\xi, \text{dip }\xi) ) ]\\
\quad [\\
  \xi . \xi (\text{sn} := \text{inc (sn }\xi) ) ]\\
\quad [\\
  \xi . \xi (\text{rreqs} := \text{rreqs }\cup \{(\text{ip }\xi, \text{sn }\xi)\} ) ]\\
\quad \text{broadcast}(\lambda \xi . \text{rreq}(0, \text{dip }\xi, \text{sqn (rt }\xi) (\text{dip }\xi), \text{sqnf (rt }\xi) (\text{dip }\xi), \text{ip }\xi, \text{sn }\xi, \text{ip }\xi, \text{False})) . \text{AODV()})"
\]

/ "Γ_{AODV} \text{PNewPkt} = labelled \text{PNewPkt} (\\
  \langle \xi. (\text{dip }\xi = \text{ip }\xi) . \text{AODV()} \rangle\\
  + (\xi. (\text{dip }\xi \neq \text{ip }\xi) )\\
  \quad [\\
    \xi . \xi (\text{store} := \text{add (data }\xi) (\text{dip }\xi) (\text{store }\xi) ) ]\\
\quad \text{AODV()})"
\]

/ "Γ_{AODV} \text{PPkt} = labelled \text{PPkt} (\\
  \langle \xi. (\text{dip }\xi = \text{ip }\xi) . \text{AODV()} \rangle\\
  + (\xi. (\text{dip }\xi \neq \text{ip }\xi) )\\
\]

\[ \langle \xi. \text{ dip } \xi \in \nu D (\text{ rt } \xi) \rangle \\
\text{unicast}(\lambda \xi. \text{ the } (\text{ nhop } (\text{ rt } \xi) (\text{ dip } \xi)), \lambda \xi. \text{ pkt}(\text{ data } \xi, \text{ dip } \xi, \text{ oip } \xi)). \text{ AODV}() \]

\[ \langle \xi. \text{ mHandler } \xi = \text{ False} \rangle \]

\[ \langle \xi. \text{ dip } \xi = \text{ ip } \xi \rangle \]

\[ \langle \xi. \text{ dip } \xi \in \nu D (\text{ rt } \xi) \rangle \\
\text{unicast}(\lambda \xi. \text{ the } (\text{ nhop } (\text{ rt } \xi) (\text{ oip } \xi)), \lambda \xi. \text{ rrep}(0, \text{ dip } \xi, \text{ sn } \xi, \text{ oip } \xi, \text{ ip } \xi)). \\
\text{broadcast}(\lambda \xi. \text{ rreq}(\text{ hops } \xi + 1, \text{ dip } \xi, \text{ dsn } \xi, \text{ dsk } \xi, \text{ oip } \xi, \text{ osn } \xi, \text{ ip } \xi, \text{ True})). \text{ AODV}() \]

\[ \langle \xi. \text{ mHandler } \xi = \text{ True} \rangle \]

\[ \langle \xi. \text{ dip } \xi \notin \nu D (\text{ rt } \xi) \rangle \\
\text{broadcast}(\lambda \xi. \text{ rerr}(\text{ dip } \xi \mapsto \text{ snq } (\text{ rt } \xi) (\text{ dip } \xi)], \text{ ip } \xi)). \text{ AODV}() \]

\[ \langle \xi. \text{ mHandler } \xi = \text{ True} \rangle \]

\[ \langle \xi. \text{ dip } \xi \notin \nu D (\text{ rt } \xi) \rangle \\
\text{broadcast}(\lambda \xi. \text{ rreq}(\text{ hops } \xi + 1, \text{ dip } \xi, \text{ dsn } \xi, \text{ dsk } \xi, \text{ oip } \xi, \text{ osn } \xi, \text{ ip } \xi, \text{ False})). \text{ AODV}() \]

\[ \langle \xi. \text{ mHandler } \xi = \text{ False} \rangle \]

\[ \langle \xi. \text{ mHandler } \xi = \text{ True} \rangle \]

\[ \langle \xi. \text{ dip } \xi \notin \nu D (\text{ rt } \xi) \rangle \\
\text{broadcast}(\lambda \xi. \text{ rerr}(\text{ dip } \xi \mapsto \text{ snq } (\text{ rt } \xi) (\text{ dip } \xi)], \text{ ip } \xi)). \text{ AODV}() \]

\[ \langle \xi. \text{ mHandler } \xi = \text{ True} \rangle \]

\[ \langle \xi. \text{ dip } \xi \notin \nu D (\text{ rt } \xi) \rangle \\
\text{broadcast}(\lambda \xi. \text{ rreq}(\text{ hops } \xi + 1, \text{ dip } \xi, \text{ dsn } \xi, \text{ dsk } \xi, \text{ oip } \xi, \text{ osn } \xi, \text{ ip } \xi, \text{ True})). \text{ AODV}() \]

\[ \langle \xi. \text{ mHandler } \xi = \text{ False} \rangle \]
(ξ. oip ξ = ip ξ)
AODV()

<table>
<thead>
<tr>
<th>ξ. oip ξ ≠ ip ξ</th>
</tr>
</thead>
</table>

(ξ. oip ξ ∈ vD (rt ξ) ∧ dip ξ ∈ vD (rt ξ))
unicast(λξ. the (nhop (rt ξ) (oip ξ)), λξ. rrep(λξ, the (dhops (rt ξ) (dip ξ)), dip ξ,
sqn (rt ξ) (dip ξ), oip ξ, ip ξ)).
AODV()

| 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. rt := invalidate (rt ξ) (dests ξ) []
| ξ. store := setRRF (store ξ) (dests ξ) []
| broadcast(λξ. rerr(dests ξ, ip ξ)).AODV()

| ξ. dests ξ ≠ Map.empty
| broadcast(λξ. rerr(dests ξ, ip ξ)). AODV()
AODV()

| ξ. dests ξ = Map.empty
| broadcast(λξ. rerr(dests ξ, ip ξ)). AODV()

)"

"ΓAODV PErr = labelled PErr (κ ξ [] dests := (λrip. case (dests ξ) rip of None ⇒ None

| ξ. 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. rt := invalidate (rt ξ) (dests 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') ∉ ctermsl (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 ∈ ctermsl (ΓAODV PPkt) ∨
                   p ∈ ctermsl (ΓAODV PRreq) ∨
                   p ∈ ctermsl (ΓAODV PRrep) ∨
                   p ∈ ctermsl (ΓAODV PRerr) ∨
                   p ∈ ctermsl (ΓAODV PRerr) ∨"
\[ \begin{align*}
\text{p} & \in \text{ctermsl} \ (\Gamma_{AODV} \text{ PPkt}) \lor \\
\text{p} & \in \text{ctermsl} \ (\Gamma_{AODV} \text{ PRreq}) \lor \\
\text{p} & \in \text{ctermsl} \ (\Gamma_{AODV} \text{ PRrep}) \lor \\
\text{p} & \in \text{ctermsl} \ (\Gamma_{AODV} \text{ PRerr})
\end{align*} \]

using \text{assms}

by (cases \text{pn}) simp_all

definition \sigma_{AODV} :: "\text{ip} \Rightarrow (\text{state} \times (\text{state}, \text{msg}, \text{pseq}, \text{pseq label}) \text{ seqp}) \text{ set}"
where "\sigma_{AODV} \ i \equiv \{(\text{aodv_init} \ i, \ \Gamma_{AODV} \text{ PAodv})\}"

abbreviation paodv :: "\text{ip} \Rightarrow (\text{state} \times (\text{state}, \text{msg}, \text{pseq}, \text{pseq label}) \text{ seqp}, \text{msg seq_action}) \text{ automaton}"
where "paodv \ i \equiv (| \text{init} = \sigma_{AODV} \ i, \text{trans} = \text{seqp_sos} \ \Gamma_{AODV})"

lemma aodv_trans: "\text{trans (paodv} \ i) = \text{seqp_sos} \ \Gamma_{AODV}"
by simp

lemma aodv_control_within [simp]: "\text{control_within} \ \Gamma_{AODV} (\text{init (paodv} \ i))"
unfolding \sigma_{AODV}_def by (rule control_withinI) (auto simp del: \Gamma_{AODV}_simps)

lemma aodv_wf [simp]: "\text{wellformed} \ \Gamma_{AODV}"
proof (rule, intro allI)
fix \text{pn} \ \text{pn}'
show "\text{call(pn')} \notin \text{stermsl} \ (\Gamma_{AODV} \ \text{pn})"
by (cases \text{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} \ \text{p}"
by (metis aodv_labels_not_empty all_not_in_conv)

lemma aodv_ex_labelE [elim]:
assumes "\forall l \in \text{labels} \ \Gamma_{AODV} \ \text{p}. \ \text{P l p}"
and "\exists p l. \text{P l p} \Longrightarrow \text{Q}"
shows "\text{Q}"
using \text{assms} by (metis aodv_ex_label)

lemma aodv_simple_labels [simp]: "\text{simple_labels} \ \Gamma_{AODV}"
proof
fix \text{pn} \ \text{p}
assume "\text{p} \in \text{subterms}(\Gamma_{AODV} \ \text{pn})"
thus "\exists l. \text{labels} \ \Gamma_{AODV} \ \text{p} = \{l\}"
by (cases \text{pn}) simp_all cong: seqp_congs | elim disjE)+
qed

lemma \sigma_{AODV}_labels [simp]: "(\xi, \text{p}) \in \sigma_{AODV} \ \text{i} \Longrightarrow \text{labels} \ \Gamma_{AODV} \ \text{p} = \{\text{PAodv}::0\}"
unfolding \sigma_{AODV}_def by simp

lemma aodv_init_kD_empty [simp]:
"(\xi, \text{p}) \in \sigma_{AODV} \ \text{i} \Longrightarrow \text{kD (rt \ \xi)} = \{\}"
unfolding \sigma_{AODV}_def kD_def by simp

lemma aodv_init_sip_not_ip [simp]: "\neg(\text{sip (aodv_init} \ i) = \text{i})" by simp

lemma aodv_init_sip_not_ip' [simp]:
assumes "(\xi, \text{p}) \in \sigma_{AODV} \ \text{i}"
shows "\text{sip} \ \xi \neq \text{ip} \ \xi"
using \text{assms} unfolding \sigma_{AODV}_def by simp

lemma aodv_init_sip_not_i [simp]:
assumes "(\xi, \text{p}) \in \sigma_{AODV} \ \text{i}"

shows "sip ξ ≠ i"
using assms unfolding σ_{AODV-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 seq_congs [cong]

Configure the main invariant tactic for AODV.
declare Γ_{AODV-simps} [cterms_env]
aodv_proc_cases [ctermsl_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

5.4 Invariant assumptions and properties

theory E_Aodv_Predicates
imports E_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 ⇒ ip"
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 dip dsn dsk oip osn sip handled.
  msg_sender (Rreq hops dip dsn dsk oip osn sip handled) = sip"
"∀ hops dip dsn oip sip.
  msg_sender (Rrep hops dip dsn oip sip) = sip"
"∀ dests sip.
  msg_sender (Rerr dests sip) = sip"
"∀ d 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 hops dip dsn dsk oip osn sip handled.
  msg_zhops (Rreq hops dip dsn dsk oip osn sip handled) = (hops = 0 → oip = sip)
  | Rrep hops dip dsn oip sip ⇒ hops = 0 → dip = sip = sipc
  | _ ⇒ True"

lemma msg_zhops_simps [simp]:
"∀ hops dip dsn dsk oip osn sip handled.
  msg_zhops (Rreq hops dip dsn dsk oip osn sip handled) = (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"
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"
lemma rreq_rrep_sn_simps [simp]:
"\( \forall \text{hops dip dsn dsnc dip oip osn sip handled}. \)
"\( \text{rreq_rrep_sn (Rreq hops dip dsn dsk oip osn sip handled) = (osn} \geq 1) \)"
"\( \forall \text{hops dip dsn oip sip}. \text{rreq_rrep_sn (Rrep hops dip dsn oip sip) = (dsn} \geq 1) \)"
"\( \forall \text{dests sip}. \text{rreq_rrep_sn (Rerr dests sip) = True} \)"
"\( \forall \text{d dip}. \text{rreq_rrep_sn (Newpkt d dip) = True} \)"
"\( \forall \text{d dip sip}. \text{rreq_rrep_sn (Pkt d dip sip) = True} \)"
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 \neq oipc →
\( \text{oipc} \in kD(crt) ∧ (\text{sqn crt oipc} > \text{osnc} ∨ \text{the (dhops crt oipc)} \leq \text{hopsc} ∧ \text{the (flag crt oipc)} = \text{val}))\)
| Rrep hopsc dipc dsnc _ ipcc ⇒ (ipcc \neq dipc →
\( \text{dipc} \in kD(crt) ∧ \text{sqn crt dipc} = \text{dsn} ∧ \text{the (dhops crt dipc)} = \text{hopsc} ∧ \text{the (flag crt dipc)} = \text{val})\)
| _ ⇒ True"
lemma rreq_rrep_fresh [simp]:
"\( \forall \text{hops dip dsn oip osn sip handled}. \)
"\( \text{rreq_rrep_fresh crt (Rreq hops dip dsn dsk oip osn sip handled) = (sip} \neq \text{oip} → \text{oip} \in kD(crt) \land (∀ \text{ripc} \in \text{dom(destsc)}. \text{ripc} \in \text{iD(crt)} ∧ (\text{destsc ripc}) = \text{sqn crt ripc}))\)
| _ ⇒ True"

lemma rerr_invalid [simp]:
"\( \forall \text{hops dip dsn oip osn sip handled}. \)
"\( \text{rerr_invalid crt (Rreq hops dip dsn dsk oip osn sip handled) = True} \)"
"\( \forall \text{hops dip dsn oip sip}. \text{rerr_invalid crt (Rrep hops dip dsn oip sip) = True} \)"
"\( \forall \text{dests sip}. \text{rerr_invalid crt (Rerr dests sip) = ( ∀ ripc} \in \text{dom(destsc)}. \text{ripc} \in \text{iD(crt)} \land \text{the (destsc ripc}) = \text{sqn crt ripc})\)
| _ ⇒ True"

definition rerr_invalid :: "rt ⇒ msg ⇒ bool"
where "rerr_invalid crt m ≡ case m of Rerr destsc _ ⇒ (∀ ripc} \in \text{dom(destsc)}. \text{ripc} \in \text{iD(crt)} \land \text{the (destsc ripc}) = \text{sqn crt ripc})\)
| _ ⇒ True"
lemma not_in_net_ips_fst_init_missing [simp]:
"\( \forall \text{hops dip dsn dsk oip osn sip handled}. \)
"\( \text{not_in_net_ips_fst_init_missing (Rreq hops dip dsn dsk oip osn sip handled) = True} \)"
"\( \forall \text{hops dip dsn oip sip}. \text{not_in_net_ips_fst_init_missing (Rrep hops dip dsn oip sip) = True} \)"
"\( \forall \text{dests sip}. \text{not_in_net_ips_fst_init_missing (Rerr dests sip) = ( ∀ ripc} \in \text{dom(destsc)}. \text{ripc} \in \text{iD(crt)} \land \text{the (destsc ripc}) = \text{sqn crt ripc})\)
| _ ⇒ True"

definition initmissing :: "(nat ⇒ state option) × 'a ⇒ (nat ⇒ state) × 'a"
where "initmissing σ = (λi. case (fst σ) i of None ⇒ aodv_init i | Some s ⇒ s, snd σ)"
lemma not_in_net_ips_fst_init_missing [simp]:
assumes "i \notin \text{net_ips} \sigma"
  shows "\text{fst (initmissing (netgmap \text{fst} \sigma))} i = \text{aodv_init} i"
using assms unfolding initmissing_def by simp

lemma \text{fst_initmissing_netgmap_pair_fst} [simp]:
  "\text{fst (initmissing (netgmap (\lambda(p, q). (\text{fst (id p)}, \text{snd (id p), q)}) s))}
  = \text{fst (initmissing (netgmap \text{fst} s))}"
unfolding initmissing_def by auto

We introduce a streamlined alternative to \text{initmissing} with \text{netgmap} to simplify invariant statements and thus facilitate their comprehension and presentation.

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

definition \text{netglobal} :: "((nat \Rightarrow \text{state}) \Rightarrow \text{bool}) \Rightarrow ((\text{state} \times 'b) \times 'c) \text{net_state} \Rightarrow \text{bool}"
where
  "\text{netglobal P} \equiv \lambda s. P (\text{default aodv_init (netlift \text{fst} s}))"
end

5.5 Quality relations between routes

theory E_Fresher
imports E_Aodv_Data
begin

5.5.1 Net sequence numbers

On individual routes
definition \text{nsqn} :: "r \Rightarrow \text{sqn}"
where
  "\text{nsqn}, \; r \equiv \text{if } \pi_4(r) = \text{val} \lor \pi_2(r) = 0 \text{ then } \pi_2(r) \text{ else } (\pi_2(r) - 1)"

lemma \text{nsqn}\_def":
  "\text{nsqn}, \; r = (\text{if } \pi_4(r) = \text{inv} \text{ then } \pi_2(r) - 1 \text{ else } \pi_2(r))"
unfolding \text{nsqn}\_def by simp

lemma \text{nsqn}\_zero [simp]:


lemma \text{nsqn}\_val [simp]:


lemma \text{nsqn}\_inv [simp]:


lemma \text{nsqn}\_lte_dsn [simp]:


On routes in routing tables
definition \text{nsqn} :: "\text{rt} \Rightarrow \text{ip} \Rightarrow \text{sqn}"
where
  "\text{nsqn} \equiv \lambda r t \text{ dip. case } \sigma_{\text{route}}(r, \text{ dip}) \text{ of } \text{None} \Rightarrow 0 \mid \text{Some } r \Rightarrow \text{nsqn},(r)"

439
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 nsqnr_r_flag_pred [simp, intro]:
 fixes dsn dsk flag hops nhip pre
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:
"∀ 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 - 1"
proof -
 from ⟨ip ∈ vD(rt)⟩ have "ip ∈ kD(rt)"
 and "the (flag rt ip) = val" by auto
 thus ?thesis ..
qued

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 ..
qued

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_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)) 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
    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)

5.5.2 Comparing routes

definition fresher :: "r ⇒ r ⇒ bool" ("(_ ⊑ _)"") [51, 51] 50
  where "fresher r r' ≡ ((nsqn r < nsqn r') ∨ (nsqn r = nsqn r' ∧ π5(r) ≥ π5(r')))"

lemma fresherI1 [intro]:
  assumes "nsqn r < nsqn r'"
  shows "r ⊑ r'"
  unfolding fresher_def using assms by simp

lemma fresherI2 [intro]:
  assumes "nsqn r = nsqn r'" and "π5(r) ≥ π5(r')"
  shows "r ⊑ r'"
  unfolding fresher_def using assms by simp

lemma fresherI [intro]:
  assumes "(nsqn r < nsqn r') ∨ (nsqn r = nsqn r' ∧ π5(r) ≥ π5(r'))"
  shows "r ⊑ r'"
  unfolding fresher_def using assms .

lemma fresherE [elim]:
  assumes "r ⊑ r'"
  and "nsqn r < nsqn r' ⟹ P r r'"
  and "nsqn r = nsqn 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'
  shows "(dsn, dsk, flag, hops, nhip) ⊑ (dsn, dsk', flag, hops, nhip')"
unfolding fresher_def by (cases flag) simp_all

5.5.3 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 ⊑_ rt2 ≡ rt_fresher i rt1 rt2"

lemma rt_fresher_def':
  "(rt1 ⊑_ rt2) = (nsqn r (the (rt1 i)) < nsqn r (the (rt2 i)) ∨
   nsqn r (the (rt1 i)) = nsqn r (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]:

442
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) ⊑ the (rt2 dip)"
and "the (rt2 dip) ⊑ 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 "π5(λip. nsqn rt dip) = π5(λip. nsqn rt' dip)"
shows "rt ≈ dip rt'"
proof
from assms(1-2,4) have dhops': "the (dhops rt' dip) ≤ the (dhops rt dip)"
by (simp add: proj5_eq_dhops)
  with assms(1-3) show "rt ⊑ dip rt'"
  by (rule rt_fresherI2)

next
from assms(1-2,4) have dhops: "the (dhops rt dip) ≤ the (dhops rt' dip)"
  by (simp add: proj5_eq_dhops)
  with assms(2,1) assms(3) [symmetric] show "rt' ⊑ dip rt"
  by (rule rt_fresherI2)
qed

lemma rt_fresh_asE [elim]:
  assumes "rt1 ≈ dip rt2"
  and "[ rt1 ⊑ dip rt2; rt2 ⊑ dip rt1 ] ⇒ 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 ≈ dip rt2"
  shows "rt1 ⊑ dip rt2"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_asD2 [dest]:
  assumes "rt1 ≈ dip rt2"
  shows "rt2 ⊑ dip rt1"
  using assms unfolding rt_fresh_as_def by simp

lemma rt_fresh_as_sym:
  assumes "rt1 ≈ dip rt2"
  shows "rt2 ≈ dip rt1"
  using assms unfolding rt_fresh_as_def by simp

lemma not_rt_fresh_asI1 [intro]:
  assumes "¬ (rt1 ⊑ dip rt2)"
  shows "¬ (rt1 ≈ dip rt2)"
  proof
    assume "rt1 ≈ dip rt2"
    hence "rt1 ⊑ dip rt2" ..
    with ¬ (rt1 ⊑ dip rt2) show False ..
  qed

lemma not_rt_fresh_asI2 [intro]:
  assumes "¬ (rt2 ⊑ dip rt1)"
  shows "¬ (rt1 ≈ dip rt2)"
  proof
    assume "rt1 ≈ dip rt2"
    hence "rt2 ⊑ dip rt1" ..
    with ¬ (rt2 ⊑ 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 "nsqn r (the (rt2 dip)) = nsqn r (the (rt1 dip))"
  using assms(3) unfolding rt_fresher_def
  by (auto simp: rt_fresher_def2 [OF ⟨dip ∈ kD(rt1)⟩ ⟨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 dsn_n dsk_n f_n hops_n nhip_n
    where rtn [simp]: "the (rt dip) = (dsn_n, dsk_n, f_n, hops_n, nhip_n)"
    by (metis prod_cases5)
  with ⟨the (dhops rt dip) ≥ 1⟩ and ⟨dip ∈ kD(rt)⟩ have "hops_n ≥ 1"
    by (metis proj5_eq_dhops projs(4))
  from ⟨dip ∈ kD(rt)⟩ rtn have [simp]: "sqn rt dip = dsn_n"
    and [simp]: "the (dhops rt dip) = hops_n"
    and [simp]: "the (flag rt dip) = f_n"
    by (simp add: sqn_def proj5_eq_dhops [symmetric]
      proj4_eq_flag [symmetric])
  from ⟨update_arg_wf r⟩ have "(dsn_n, dsk_n, f_n, hops_n, nhip_n) ⊆ the ((update rt dip r) dip)"
    by (rule wf_r_cases)
  fix nhip
  from ⟨hops_n ≥ 1⟩ have "(dsn_n, dsk_n, f_n, hops_n, nhip_n) ⊆ (dsn_n, unk, val, Suc 0, nhip)"
unfolding fresher_def sqn_def by (cases fn) auto
thus "(dsn_n, dsk_n, fn, hops_n, nhip_n)
 ⊑ 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 "(dsn, dsk_n, fn, hops_n, nhip_n)
 ⊑ (dsn, kno, val, hops, nhip)"
proof (rule update_cases_kD [OF ⟨dip∈kD(rt)⟩], simp_all)
unfolding fresher_def by auto

next
assume "dsn = dsn"
and "hops < hops_n"
thus "(dsn, dsk_n, fn, hops_n, nhip_n)
 ⊑ (dsn, kno, val, hops, nhip)"
unfolding fresher_def nsqn_r_def by simp

next
assume "dsn = dsn"
with ⟨0 < dsn⟩
show "(dsn, dsk_n, inv, hops_n, nhip_n)
 ⊑ (dsn, kno, val, hops, nhip)"
unfolding fresher_def by simp
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 "nsqn_r (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∈vD(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 = nsqn r (the (invalidate rt dests dip))"
    proof
      -
      from ⟨dip∈kD(rt)⟩ have "nsqn r (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 = nsqn r (the (invalidate rt dests dip))" by simp
    qed
  also from ⟨dip∈kD(invalidate rt dests)⟩
  have "nsqn r (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 ⊏ rt2" ≡ rt_strictly_fresher i rt1 rt2"
lemma rt_strictly_fresher_def'':
  "rt1 ⊏ rt2 = ((rt1 ⊑ rt2) ∧ ¬(rt2 ⊑ rt1))"
unfolding rt_strictly_fresher_def rt_fresh_as_def by auto
lemma rt_strictly_fresherI' [intro]:
  assumes "rt1 ⊑ rt2"
  and "¬(rt2 ⊑ rt1)"
  shows "rt1 ⊏ rt2"
  using assms unfolding rt_strictly_fresher_def'' by simp
lemma rt_strictly_fresherE' [elim]:
  assumes "rt1 ⊑ rt2"
  and "[ rt1 ⊑ rt2; ¬(rt2 ⊑ 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 ⊑ rt2"
  and "¬(rt1 ≈ rt2)"
  shows "rt1 ⊏ 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 ⊑ rt2"
and "[ rt1 ⊑ rt2; ¬(rt1 ≈ 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 =
(nsqr_r (the (rt1 i)) < nsqr_r (the (rt2 i))
 ∨ (nsqr_r (the (rt1 i)) = nsqr_r (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 ⊏ dip rt2"
shows "the (rt1 dip) ⊑ 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 ⊏ dip rt2"
shows "¬ rt1 ≈ dip rt2"
using assms unfolding rt_strictly_fresher_def by auto

lemma rt_strictly_fresher_trans [elim, trans]:
assumes "rt1 ⊏ dip rt2"
and "rt2 ⊑ dip rt3"
shows "rt1 ⊏ dip rt3"
using assms proof -
from rt1 ⊏ dip rt2: obtain "the (rt1 dip) ⊑ the (rt2 dip)" by auto
also from rt2 ⊑ dip rt3: obtain "the (rt2 dip) ⊑ the (rt3 dip)" by auto
finally have "the (rt1 dip) ⊑ the (rt3 dip)" .
moreover have "¬ (rt1 ≈ dip rt3)"
proof -
from rt1 ⊏ dip rt2: obtain "¬(the (rt2 dip) ⊑ the (rt1 dip))" by auto
also from rt2 ⊏ dip rt3: obtain "¬(the (rt3 dip) ⊑ the (rt2 dip))" by auto
finally have "¬(the (rt3 dip) ⊑ the (rt1 dip))" .
thus ?thesis ..
qed ultimately show "rt1 ⊏ dip rt3" ..
qed

lemma rt_strictly_fresher_irefl [simp]: "¬ (rt ⊏ dip rt)"
using assms unfolding rt_strictly_fresher_def by clarsimp

lemma rt_fresher_trans_rt_strictly_fresher [elim, trans]:
assumes "rt1 ⊏ dip rt2"
and "rt2 ⊏ dip rt3"
shows "rt1 ⊏ 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_nwil)
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"
  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
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 "nsqn, (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 = osn" by (rule sqn [symmetric])
  also have "sqn (rt2 nhip) dip = nsqn, (the (rt2 nhip dip))"
    unfolding nsqn_def using ⟨dip ∈ vD (rt2 nhip)⟩
    by (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show "nsqn, (the (update rt dip (osn, kno, val, hops, nhip) dip)) < nsqn, (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 "nsqn, (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 = nsqn, (the (rt2 nhip dip))"
    unfolding nsqn_def using ⟨dip ∈ vD (rt2 nhip)⟩
    by (metis vD_flag_val proj2_eq_sqn proj4_eq_flag vD_iD_gives_kD(1))
  finally show "nsqn, (the (update rt dip (osn, kno, val, Suc hops, nhip) dip)) = nsqn, (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 "π 5 (the (rt2 nhip dip)) < π 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 "∀ ip ∈ dom(dests). ip ∈ vD(rt) ∧ the (dests ip) = inc (sn 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

5.6 Invariant proofs on individual processes

theory E_Seq_Invariants
imports "../../../AWN/Invariants" E_Aodv E_Aodv_Data E_Aodv_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:
"paodv i ‹|=› onll Γ AODV (λ((ξ, _), _, (ξ', _)). sn ξ ≤ sn ξ')"
by inv_cterms

lemma sequence_number_one_or_bigger:
"paodv i ‹|=› onl Γ AODV (λ(ξ, _). 1 ≤ sn ξ)"
by (rule onll_step_to_invariantI [OF sequence_number_increases])
(auto simp: σ AODV_def)

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

lemma sequence_number_increases':
"paodv i ‹|=› (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 Γ AODV (λ(ξ, 1). l ∈ {{PAodv-:7} ⊔ {PAodv-:5} ⊔ {PRrep-:0..PRrep-:4}
∪ {PRreq-:0..PRreq-:3}} →→ sip ξ ∈ kD (rt ξ))"
by inv_cterms

Proposition 7.38

lemma includes_nhip:
"paodv i ‹|=› onl Γ AODV (λ(ξ, 1). ∀ 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 := 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 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))"
  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 aodv_wf sip_in_kD]
  solve: one_hop nhip_is_sip)

qed

Proposition 7.4
lemma known_destinations_increase:
"paodv i |= A onll Γ AODV (λ((ξ, _), _, (ξ', _)). kD (rt ξ) ⊆ kD (rt ξ'))"
by (inv_cterms simp add: subset_insertI)

Proposition 7.5
lemma rreqs_increase:
"paodv i |= A onll Γ AODV (λ((ξ, _), _, (ξ', _)). rreqs ξ ⊆ rreqs ξ')"
by (inv_cterms simp add: subset_insertI)

lemma dests_bigger_than_sqn:
"paodv i |= onl Γ AODV (λ(ξ, 1). l ∈ {PAodv-:15..PAodv-:17}
  ∪ {PPkt-:7..PPkt-:9}
  ∪ {PRreq-:11..PRreq-:13}
  ∪ {PRreq-:20..PRreq-:22}
  ∪ {PRrep-:7..PRrep-:9}
  ∪ {PRerr-:1..PRerr-:4} ∪ {PRerr-:6}
  → (∀ 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:
"paodv i |= A onll Γ AODV (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 aodv_wf dests_bigger_than_sqn]
    simp add: solve_invalidate)
qed

Proposition 7.7

lemma ip_constant:
"paodv i |= onl ΓAODV (λ(ξ, _). ip ξ = i)"
by (inv_cterms simp add: σAODV_σ_def)

Proposition 7.8

lemma sender_ip_valid':
"paodv 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:
"paodv 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:
"paodv i |= (recmsg P −→) onl ΓAODV (λ(ξ, 1). l ∈ {PAodv←:1} −→ P (msg ξ))"
by inv_cterms

Proposition 7.9

lemma sip_not_ip':
"paodv i |= (recmsg (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 aodv_wf received_msg_inv]
    onl_invariant_sterms [OF aodv_wf ip_constant [THEN invariant_restrict_inD]]
    simp add: clear_locals_sip_not_ip') clarsimp+

lemma sip_not_ip:
"paodv i |= (recmsg (λm. not_Pkt m −→ msg_sender m = i) →) onl ΓAODV (λ(ξ, _). sip ξ = i)"
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf received_msg_inv]
    onl_invariant_sterms [OF aodv_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:
"paodv i |= onl ΓAODV (λ(ξ, _). ∀ip∈kD (rt ξ). the (dhops (rt ξ) ip) ≥ 1)"
by (inv_cterms) auto

lemma rreq_dip_in_vD_dip_eq_ip:
"paodv i |= onl ΓAODV (λ(ξ, 1). l ∈ {PRreq←:16..PRreq←:17} −→ dip ξ ∈ vD(rt ξ))
∧ (l ∈ {PRreq←:6, PRreq←:7} −→ dip ξ = ip ξ)
∧ (l ∈ {PRreq←:15..PRreq←:17} −→ dip ξ = ip ξ))"
by inv_cterms

lemma rrep_dip_in_vD:
"paodv i |= onl ΓAODV (λ(ξ, 1). l ∈ {PRrep←:4} −→ dip ξ ∈ vD(rt ξ))"
by inv_cterms

Proposition 7.11

lemma anycast_msg_zhops:
"\forall rreqid dip dsn dsk oip osn sip.
   paodv i |=A onll ΓAODV (λ(ξ, a, _). anycast msg_zhops a)"
proof (inv_cterms inv add:
  onl_invariant_sterms [OF aodv_wf rreq_dip_in_vD_dip_eq_ip]
  onl_invariant_sterms [OF aodv_wf rrep_dip_in_vD]
  onl_invariant_sterms [OF aodv_wf hop_count_positive],
  elim conjE)

fix l ξ a pp p' pp'
assume "((ξ, pp) ∈ reachable (paodv i) TT"
  and "({PRreq-:16}unicast(λξ'. the (nhop (rt ξ) (oip ξ)),
     ξ'. Rrep (the (dhops (rt ξ) (dip ξ))) (dip ξ) (sqn (rt ξ) (dip ξ)) (oip ξ) (ip ξ)).
     p' > pp' ∈ sterms Γ_AODV pp"
  and "1 = PRreq-:16"
  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_1D_gives_kD(1))
with * have "Suc 0 ≤ the (dhops (rt ξ) (dip ξ))" ..
thus "0 < the (dhops (rt ξ) (dip ξ))" by simp

next
fix l ξ a pp p' pp'
assume "((ξ, pp) ∈ reachable (paodv i) TT"
  and "({PRreq-:4}unicast(λξ'. the (nhop (rt ξ) (oip ξ)),
     ξ'. Rrep (the (dhops (rt ξ) (dip ξ))) (dip ξ) (sqn (rt ξ) (dip ξ)) (oip ξ) (ip ξ)).
     p' > pp' ∈ sterms Γ_AODV pp"
  and "1 = PRreq-:4"
  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_1D_gives_kD(1))
with * have "Suc 0 ≤ the (dhops (rt ξ) (dip ξ))" ..
thus "the (dhops (rt ξ) (dip ξ)) = 0 → dip ξ = ip ξ" by auto

qed

lemma hop_count_zero_oip_dip_sip:
  "paodv i |= (recvmsg msg_zhops →) onl Γ_AODV (λ(ξ, 1).
    (1∈{PAodv-:4, PAodv-:5} ∨ {PRreq-:n|n. True} →
      (hops ξ = 0 → oip ξ = sip ξ))
    ∧
    (l∈{PAodv-:4, PAodv-:5} ∨ {PRreq-: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 |= (recvmsg rreq_rrep_sn →) onl Γ_AODV (λ(ξ, 1).
    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 |= (recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) →) onl Γ_AODV (λ(ξ, 1).
    l ∈ {PAodv-:4, PAodv-:5} ∨ {PRreq-:n|n. True} → 1 ≤ osn ξ)"
proof (rule invariant_weakenE [OF osn_rreq])
  fix a
  assume "recvmsg (λm. rreq_rrep_sn m ∧ msg_zhops m) a"
  thus "recvmsg rreq_rrep_sn a" by (cases a) simp_all

qed

lemma dsn_rrep:
  "paodv i |= (recvmsg rreq_rrep_sn →) onl Γ_AODV (λ(ξ, 1).
    l ∈ {PAodv-:6, PAodv-:7} ∨ {PRreq-:n|n. True} → 1 ≤ dsn ξ)"
lemma dsn_rrep'::
"{paodv i} ||= (recvmsg (\m. rreq_rrep_sn m \& msg_zhops m) \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, 1).
  \{\{PAodv-:6..PAodv-:7\} \cup \{PRrep-:n|n. True\} \rightarrow 1 \leq dsn \xi\})"
proof (rule invariant_weakenE [OF dsn_rrep])
  fix a
  assume "recvmsg (\m. rreq_rrep_sn m \& msg_zhops m) a"
  thus "recvmsg rreq_rrep_sn a"
  by (cases a) simp_all
qed

lemma hop_count_zero_oip_dip_sip'::
"{paodv i} ||= (recvmsg (\m. rreq_rrep_sn m \& msg_zhops m) \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, _).
  \{\{PAodv-:4..PAodv-:5\} \cup \{PRreq-:n|n. True\} \rightarrow (hops \xi = 0 \rightarrow oip \xi = sip \xi)\}
  \& \{\{PAodv-:6..PAodv-: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 (\m. rreq_rrep_sn m \& msg_zhops m) a"
  thus "recvmsg msg_zhops a"
  by (cases a) simp_all
qed

Proposition 7.12
lemma zero_seq_unk_hops_one'::
"{paodv i} ||= (recvmsg (\m. rreq_rrep_sn m \& msg_zhops m) \rightarrow) onl \Gamma_{AODV} (\lambda(\xi, _).
  \forall dip \& \xi :: state and P
  \exists 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 dip and \xi :: state and P
  assume "\forall dip \& \xi :: state
  (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 \mathbb{D}(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!: bepec)
} note prreq_ok1 [simp] = this

{ fix ip dip and \xi :: state and P
  assume "\forall dip \& \xi :: state
  (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 \mathbb{D}(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!: bepec)
} note prreq_ok1 [simp] = this
and \[ ip \neq dip \rightarrow ip \in kD(rt) \]
hence \[ \pi_3(\text{the (update rt dip (dsn, kno, Suc hops, sip) ip)) = unk} \rightarrow \]
the (dhops (update rt dip (dsn, kno, Suc hops, sip) ip)) = Suc 0
by (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)

{ fix ip dsn hops sip oip rt dip
assume \[ \forall dip \in kD(rt). \]
\[ (sqn rt dip = 0 \rightarrow \pi_3(\text{the (rt dip)}) = unk) \land \]
\[ (\pi_3(\text{the (rt dip)}) = unk \rightarrow \text{the (dhops rt dip) = Suc 0}) \land \]
\[ \text{the (dhops rt dip) = Suc 0} \rightarrow \text{the (nhop rt dip) = dip}) \]
and \[ "Suc 0 \leq dsn"\]
and \[ "ip \neq dip \rightarrow ip \in kD(rt)"\]
hence \[ sqn (update rt dip (dsn, kno, Suc hops, sip)) ip = 0 \rightarrow \]
\[ \pi_3(\text{the (update rt dip (dsn, kno, Suc hops, sip) ip)) = unk} \]
by (rule update_cases, auto simp add: sqn_def sqnf_def dest!: bspec)
}

{ fix rt sip
assume \[ \forall dip \in kD rt. \]
\[ (sqn rt dip = 0 \rightarrow \pi_3(\text{the (rt dip)}) = unk) \land \]
\[ (\pi_3(\text{the (rt dip)}) = unk \rightarrow \text{the (dhops rt dip) = Suc 0}) \land \]
\[ \text{the (dhops rt dip) = Suc 0} \rightarrow \text{the (nhop rt dip) = dip}) \]
hence \[ "sqn (update rt sip (0, unk, val, Suc 0, sip)) dip = 0 \rightarrow \]
\[ \pi_3(\text{the (update rt sip (0, unk, val, Suc 0, sip) dip)) = unk} \]
\[ \text{the (dhops (update rt sip (0, unk, val, Suc 0, sip)) dip) = Suc 0} \]
\[ \text{the (nhop (update rt sip (0, unk, val, Suc 0, sip)) dip) = dip}) \]
by (rule update_cases, simp_all add: sqn_def sqnf_def)
}

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

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

show ?thesis
by (inv_cterms inv add: onl_invariant_sterms [OF aodv_wf hop_count_zero_oip_dip_sip']
seq_step_invariant_sterms_TT [OF sqns_increase aodv_wf aodv_trans]
onl_invariant_sterms [OF aodv_wf osn_rreq']
onl_invariant_sterms [OF aodv_wf dsn_rrep']) clarsimp+

qed

lemma zero_seq_unk_hops_one:
"paodv i |\langle recvmsg (\lambda m. rreq_rrep_sn m \land 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 \land \text{the (dhops (rt \xi) dip) = 1}) \land \text{the (nhop (rt \xi) dip) = dip}))"

by (rule invariant_weakenE [OF zero_seq_unk_hops_one']) auto

lemma kD_unk_or_atleast_one:
"paodv i |\langle recvmsg rreq_rrep_sn |\rightarrow onl \Gamma_{AODV} (\lambda(\xi, 1). \]
\forall dip \in kD(rt \xi). \pi_3(\text{(the (rt dip)) = unk \lor 1 \leq \pi_2(\text{(the (rt dip))})))"

proof -
{ fix sip rt dsn1 dsn2 dsk1 dsk2 flag1 flag2 hops1 hops2 nhp1 nhp2
assume \[ "dsk1 = unk \lor Suc 0 \leq dsn2"\]
hence \[ "\pi_3(\text{(the (update rt sip (dsn1, dsk1, flag1, hops1, nhp1) sip)) = unk}} \]
Suc 0 ≤ sqn (update rt sip (dsn1, dsk1, flag1, hops1, nhip1)) sip" unfolding update_def by (cases "dsk1 = unk") (clarsimp split: option.split)+

\[
\forall \text{ dip } \in \text{kD}(rt). \pi_3(\text{the (rt dip)}) = \text{unk} \lor \text{Suc 0} \leq \text{sqn rt dip}
\]

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

(is "\forall \text{ dip } \in \text{kD}(rt). \text{?prop dip}")

proof
fix dip assume "\text{dip} \in \text{kD}(rt)"
thus "\text{?prop dip}"
proof (cases "\text{dip} = \text{sip}"
assume "\text{dip} = \text{sip}"
with ** show ?thesis
by simp
next assume "\text{dip} \neq \text{sip}"
with ⟨\text{dip} \in \text{kD}(rt)⟩ allkd show ?thesis
by simp
qed
qed

\[
\forall \text{ dip } \in \text{kD}(rt). \pi_3(\text{the (update rt sip (dsn1, dsk1, flag1, hops1, nhip1) dip)}) = \text{unk} \lor \text{Suc 0} \leq \text{sqn (update rt sip (dsn2, dsk2, flag2, hops2, nhip2)) dip}
\]

proof
fix dip assume "\text{dip} \in \text{kD}(rt)"
thus "\text{?prop dip}"
proof (cases "\text{dip} = \text{sip}"
assume "\text{dip} = \text{sip}"
with ** show ?thesis
by simp
next assume "\text{dip} \neq \text{sip}"
with ⟨\text{dip} \in \text{kD}(rt)⟩ allkd show ?thesis
by simp
qed
qed

\[
\forall \text{ dip } \in \text{kD}(rt). \text{?prop dip}
\]

proof
fix dip assume "\text{dip} \in \text{kD}(rt)"
thus "\text{?prop dip}"
proof (cases "\text{dip} = \text{sip}"
assume "\text{dip} = \text{sip}"
with ** show ?thesis
by simp
next assume "\text{dip} \neq \text{sip}"
with ⟨\text{dip} \in \text{kD}(rt)⟩ allkd show ?thesis
by simp
qed
qed

\[
\forall \text{ dip } \in \text{kD}(rt). \text{?prop dip}
\]

proof
fix dip assume "\text{dip} \in \text{kD}(rt)"
thus "\text{?prop dip}"
proof (cases "\text{dip} = \text{sip}"
assume "\text{dip} = \text{sip}"
with ** show ?thesis
by simp
next assume "\text{dip} \neq \text{sip}"
with ⟨\text{dip} \in \text{kD}(rt)⟩ allkd show ?thesis
by simp
qed
qed

\[
\forall \text{ dip } \in \text{kD}(rt). \text{?prop dip}
\]

proof
fix dip assume "\text{dip} \in \text{kD}(rt)"
thus "\text{?prop dip}"
proof (cases "\text{dip} = \text{sip}"
assume "\text{dip} = \text{sip}"
with ** show ?thesis
by simp
next assume "\text{dip} \neq \text{sip}"
with ⟨\text{dip} \in \text{kD}(rt)⟩ allkd show ?thesis
by simp
qed
qed

\[
\forall \text{ dip } \in \text{kD}(rt). \text{?prop dip}
\]

proof
fix dip assume "\text{dip} \in \text{kD}(rt)"
thus "\text{?prop dip}"
proof (cases "\text{dip} = \text{sip}"
assume "\text{dip} = \text{sip}"
with ** show ?thesis
by simp
next assume "\text{dip} \neq \text{sip}"
with ⟨\text{dip} \in \text{kD}(rt)⟩ allkd show ?thesis
by simp
qed
qed

\[
\forall \text{ dip } \in \text{kD}(rt). \text{?prop dip}
\]

proof
fix dip assume "\text{dip} \in \text{kD}(rt)"
thus "\text{?prop dip}"
proof (cases "\text{dip} = \text{sip}"
assume "\text{dip} = \text{sip}"
with ** show ?thesis
by simp
next assume "\text{dip} \neq \text{sip}"
with ⟨\text{dip} \in \text{kD}(rt)⟩ allkd show ?thesis
by simp
qed
qed

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:

"paodv i \|=_{A} (recvmsg rreq_rrep_sn \rightarrow) onll \Gamma_{AODV} (\lambda(_, a, _). anycast rreq_rrep_sn a)"

proof -

have sqnf_kno: "paodv i \|=_{A} onll \Gamma_{AODV} (\lambda(\xi, l). anycast (rreq_rrep_sn (rt \xi)))"

by (inv_cterms)

have rrep_sqn_greater_dsn: "paodv i \|=_{A} (recvmsg rreq_rrep_sn \rightarrow) onll \Gamma_{AODV} (\lambda(_, a, _). anycast rreq_rrep_sn a)"

proof -

have rreq_oip: "paodv i \|=_{A} onll \Gamma_{AODV} (\lambda(\xi, l). anycast (rreq_rrep_fresh (rt \xi)))"

proof inv_cterms

fix \xi \ l' pp p'

assume "((\xi, pp) \in reachable (paodv i))"

and "\{PRreq:-3..PRreq:-9\} \cup \{PRreq:-15, PRreq:-24, PRreq:-26\} \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

fix \xi \ l' pp p'

assume "((\xi, pp) \in reachable (paodv i))"

and "\{PRreq:-2\}\{\xi. \ l := update (rt \xi) (oip \xi) (osn \xi, kno, val, Suc (hops \xi), sip \xi)\} pp'

by (inv_cterms)

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 option.split)

(auto simp: proj2_eq_sqn)

qed

Proposition 7.14

lemma rreq_rrep_fresh_any_step_invariant:

"paodv i \|=_{A} onll \Gamma_{AODV} (\lambda(\xi, l). anycast (rreq_rrep_fresh (rt \xi)))"

proof -

have rreq_oip: "paodv i \|=_{A} onll \Gamma_{AODV} (\lambda(\xi, l). anycast (rreq_rrep_fresh (rt \xi)))"

proof inv_cterms

fix \xi \ l' pp p'

assume "((\xi, pp) \in reachable (paodv i))"

and "\{PRreq:-3..PRreq:-9\} \cup \{PRreq:-15, PRreq:-24, PRreq:-26\} \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)))"
\[ (\text{sqn } (\text{rt } \xi) (\text{oip } \xi) = (\text{osn } \xi)) \\
\wedge \text{the } (\text{dhops } (\text{rt } \xi) (\text{oip } \xi)) \leq \text{Suc } (\text{hops } \xi) \\
\wedge \text{the } (\text{flag } (\text{rt } \xi) (\text{oip } \xi) = \text{val})) ) \] 

by (\text{inv_cterms inv add: onl_invariant_sterms [OF aodv_wf rreq_oip]})

show ?thesis

qed

**Proposition 7.15**

**Lemma rerr_invalid_any_step_invariant:**

"paodv i \vdash \text{onll } \Gamma_{AODV} (\lambda((\xi, _), a, _). \text{anycast } (\text{rerr_invalid } (\text{rt } \xi)) a)"

proof -

have dests_inv: "paodv i \vdash \text{onll } \Gamma_{AODV} (\lambda((\xi, l), a, _). (l \in \{\text{PAodv-:15, PPkt-:7, PRreq-:11, PRreq-:20, PRrep-:7, PRerr-:1}\} \\
\wedge (l \in \{\text{PAodv-:16..PAodv-:17}\} \cup \{\text{PPkt-:8..PPkt-:9}\} \\
\cup \{\text{PRreq-:12..PRreq-:13}\} \cup \{\text{PRreq-:21..PRreq-:22}\} \\
\cup \{\text{PRrep-:8..PRrep-:9}\} \cup \{\text{PRerr-:2..PRerr-:4}\} \\
\wedge (1 = \text{PPkt-:12} \rightarrow \text{dip } \xi \in \text{I}(\text{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 aodv_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 \((\text{rip}, \text{rsn})\) is assigned to the variable \(\text{dests}\) of type \(\text{ip } \rightarrow \text{sqn}\), or to the first argument of the function \(\text{rerr}\), this set is a partial function, i.e., there is at most one entry \((\text{rip}, \text{rsn})\) for each destination \(\text{rip}\): guaranteed by type.

**Lemma dests_vD_inc_sqn:**

"paodv i \vdash \text{onll } \Gamma_{AODV} (\lambda((\xi, l). (l \in \{\text{PAodv-:15, PPkt-:7, PRreq-:11, PRreq-:20, PRrep-:7}\} \\
\wedge (1 \in \{\text{PAodv-:16..PAodv-:17}\} \cup \{\text{PPkt-:8..PPkt-:9}\} \\
\cup \{\text{PRreq-:12..PRreq-:13}\} \cup \{\text{PRreq-:21..PRreq-:22}\} \\
\cup \{\text{PRrep-:8..PRrep-:9}\} \cup \{\text{PRerr-:2..PRerr-:4}\} \\
\wedge (1 = \text{PPkt-:12} \rightarrow \text{dip } \xi \in \text{I}(\text{rt } \xi)))"

by inv_cterms (clarsimp split: split_if_asm option.split_asm simp: domIff)+

**Proposition 7.27**

**Lemma route_tables_fresher:**

"paodv i \vdash (\text{recvmsg } \text{rreq_rrep_sn} \rightarrow \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _). (\xi'))" \\
\forall \text{dip } \in \text{D}(\text{rt } \xi). \text{rt } \xi \in \text{dip } \text{rt } \xi')"

proof (\text{inv_cterms inv add: onl_invariant_sterms [OF aodv_wf dests_vD_inc_sqn [THEN invariant_restrict_inD]] onl_invariant_sterms [OF aodv_wf hop_count_positive [THEN invariant_restrict_inD]] onl_invariant_sterms [OF aodv_wf osn_rreq] onl_invariant_sterms [OF aodv_wf dsn_rrep])
5.7 The quality increases predicate

theory E_Quality_Increases
imports E_Aodv_Predicates E_Fresher
begin

definition quality_increases :: "state \⇒ bool" where "quality_increases ξ ξ' ≡ (\forall dip \in kD(\xi). dip \in kD(\xi') \∧ rt ξ \sqsubseteq dip rt ξ') \∧ (\forall sqn (rt ξ) dip \leq sqn (rt ξ') dip)"

lemma quality_increasesI [intro!]:
  assumes "\\forall dip. dip \in kD(\xi) \⇒ dip \in kD(\xi')"
  and "\\forall dip. dip \in kD(\xi) \⇒ rt ξ \sqsubseteq dip rt ξ'"
  and "\\forall dip. sqn (rt ξ) dip \leq 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 \in kD(\xi)"
  and "[ dip \in kD(\xi'); rt ξ \sqsubseteq dip rt ξ'; sqn (rt ξ) dip \leq sqn (rt ξ') dip ] \⇒ R dip ξ'"
  shows "R dip ξ'"
using assms unfolding quality_increases_def by clarsimp

lemma quality_increases_rt_fresherD [dest]:
fixes ip
assumes "quality_increases ξ 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 \implies 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 \in 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 \in kD(rt (σ nhip))⟩
  by auto
  with ⟨rt (σ i) ▦ dip rt (σ nhip)⟩ show ?thesis ..
qed

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

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

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

lemma kD_nsqn_quality_increases_trans [elim]:
  assumes "i \in kD(rt ξ)"
  and "s \leq nsqn (rt ξ) i"
  and "quality_increases ξ ξ'"
  shows "i \in kD(rt ξ') \land s \leq nsqn (rt ξ') i"
proof
  from ⟨i \in kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ show "i \in kD(rt ξ')" ..
  next
  from ⟨i \in kD(rt ξ)⟩ and ⟨quality_increases ξ ξ'⟩ have "nsqn (rt ξ) i \leq nsqn (rt ξ') i" ..
  with ⟨s \leq nsqn (rt ξ) i⟩ show "s \leq 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 ..
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

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" ..

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

thus ?thesis by (metis (mono_tags) le_cases not_le)
qed

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

\(\text{qed}\)

\[\]

**Lemma rteq\_quality\_increases:**

assumes \(\forall j. j \neq i \rightarrow \text{quality\_increases} (\sigma j) (\sigma' j)\)

and \(\text{rt} (\sigma' i) = \text{rt} (\sigma i)\)

shows \(\forall j. \text{quality\_increases} (\sigma j) (\sigma' j)\)

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

**Definition msg\_fresh :: \((ip \Rightarrow \text{state}) \Rightarrow \text{msg} \Rightarrow \text{bool}\)**

where \(\text{msg\_fresh} \ \sigma m \equiv\)

\[
\begin{align*}
\text{case m of} & \\
\text{Rreq} \ \text{hopsc} \ _ _ \ _ _ \ oipc \ osnc \ sipc \ _ _ \Rightarrow & \text{oipc} \in \text{kD(\text{rt} (\sigma \ sipc))) \land \text{nsqn (rt (\sigma sipc)) oipc} \geq \text{osnc} \\
& \rightarrow \text{(hopsc} \geq \text{the (dhops (rt (\sigma sipc))) oipc)} \\
& \lor \text{the (flag (rt (\sigma sipc)) oipc) = inv)) \\
\text{Rrep} \ \text{dipc} \ dsnc \ _ _ \ sipc \Rightarrow & \text{dsnc} \geq 1 \land \text{nsqn (rt (\sigma sipc)) dsnc} \geq \text{dsn} \\
& \rightarrow \text{(dsnc} \geq \text{the (dhops (rt (\sigma sipc))) dsnc)} \\
& \lor \text{the (flag (rt (\sigma sipc)) dsnc) = inv)) \\
\text{Rerr} \ \text{destsc} \ sipc \Rightarrow & \forall \text{ripc} \in \text{dom(destsc). (ripc} \in \text{kD(\text{rt} (\sigma \ sipc))} \\
& \land \text{nsqn (rt (\sigma sipc)) ripc} \geq \text{osnc} \\
& \lor \text{the (destsc ripc) - 1} \leq \text{nsqn (rt (\sigma sipc)) ripc)}
\end{align*}
\]

**Lemma msg\_fresh \[simpl\]:**

"\(\forall \text{hops dip dsn dsk oip osn sip handled.}
\text{msg\_fresh} \ \sigma \ (\text{Rreq} \ \text{hops dip dsn dsk oip osn sip handled}) =\)

\[
\begin{align*}
& (\text{osn} \geq 1 \land (\text{ripc} \notin oip \rightarrow \text{oipc} \in \text{kD(\text{rt} (\sigma \ sipc)))} \\
& \land \text{nsqn (rt (\sigma sipc)) oipc} \geq \text{osnc} \\
& \land (\text{nsqn (rt (\sigma sipc)) oipc} = \text{osnc} \\
& \rightarrow \text{(hops} \geq \text{the (dhops (rt (\sigma sipc))) oipc)} \\
& \lor \text{the (flag (rt (\sigma sipc)) oipc) = inv))\))
\end{align*}
\]

"\(\forall \text{hops dip dsn oip sip. msg\_fresh} \ \sigma \ (\text{Rrep} \ \text{hops dip dsn oip sip}) =\)

\[
\begin{align*}
& (\text{dsn} \geq 1 \land (\text{ripc} \notin dip \rightarrow \text{dipc} \in \text{kD(\text{rt} (\sigma \ sipc)))} \\
& \land \text{nsqn (rt (\sigma sipc)) dip} \geq \text{dsn} \\
& \land (\text{nsqn (rt (\sigma sipc)) dip} = \text{dsn} \\
& \rightarrow \text{(hops} \geq \text{the (dhops (rt (\sigma sipc))) dip)} \\
& \lor \text{the (flag (rt (\sigma sipc)) dip) = inv))\))
\end{align*}
\]

"\(\forall \text{dests sip.}
\text{msg\_fresh} \ \sigma \ (\text{Rerr} \ \text{dests sip}) =\)

\[
\begin{align*}
& (\forall \text{ripc} \in \text{dom(dests). (ripc} \in \text{kD(\text{rt} (\sigma \ sipc)))} \\
& \land \text{nsqn (rt (\sigma sipc)) ripc} \geq \text{osnc} \\
& \lor \text{the (dests ripc) - 1} \leq \text{nsqn (rt (\sigma sipc)) ripc)}
\end{align*}
\]

"\(\forall \text{d dip.}
\text{msg\_fresh} \ \sigma \ (\text{Newpkt} \ d dip) = \text{True}\"  \
"\(\forall \text{d dip sip.}
\text{msg\_fresh} \ \sigma \ (\text{Pkt} \ d dip sip) = \text{True}\"

unfolding msg\_fresh\_def by simp\_all

**Lemma msg\_fresh\_inc\_sn \[simpl, elim\]:**

"msg\_fresh \ \sigma m \Rightarrow \text{rreq\_rrep\_sn m}"

by (cases m) simp\_all

**Lemma recv\_msg\_fresh\_inc\_sn \[simpl, elim\]:**

"orecvmsg (msg\_fresh) \ \sigma m \Rightarrow \text{recvmsg rreq\_rrep\_sn m}"

by (cases m) simp\_all

**Lemma rreq\_nsqn\_is\_fresh \[simpl\]:**

fixes \(\sigma \ \text{msg hops dip dsn dsk oip osn sip handled}\)

assumes "rreq\_rrep\_fresh (rt (\sigma sipc)) (\text{Rreq} \ \text{hops dip dsn dsk oip osn sip handled})"

and "rreq\_rrep\_sn (\text{Rreq} \ \text{hops dip dsn dsk oip osn sip handled})"

shows "msg\_fresh \ \sigma \ (\text{Rreq} \ \text{hops dip dsn dsk oip osn sip handled})"

(is "msg\_fresh \ \sigma \ ?msg")

proof -

let \(\text{?rt = "rt (\sigma sipc)"}

from \(assms(2)\) have "1 \leq \text{osn}" by simp

thus \(?thesis\)
unfolding \texttt{msg\_fresh\_def}
proof (simp only: msg.case, intro conjI impI)
  assume "sip ≠ oip"
  with \texttt{assms(1)} show "oip ∈ kD(?rt)" by \texttt{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 \texttt{nsqn ?rt oip = osn} have "sqn ?rt oip = osn" by \texttt{simp}
  with \texttt{assms(1)} and \texttt{sip ≠ oip} have "the (dhops ?rt oip) ≤ hops"
    by \texttt{simp}
  thus ?thesis ..
next
  assume "oip∈vD(?rt)"
  moreover from \texttt{assms(1)} and \texttt{sip ≠ oip} have "oip∈kD(?rt)" by \texttt{simp}
  ultimately have "oip∈iD(?rt)" by \texttt{auto}
  hence "the (flag ?rt oip) = inv" ..
  thus ?thesis ..
qed

lemma \texttt{rrep\_nsqn\_is\_fresh [simp]}:
  \texttt{fixes \sigma msg hops dip dsn oip sip}
  \texttt{assumes "rreq\_rrep\_fresh (rt (\sigma sip)) (Rrep hops dip dsn oip sip)"
    and "rreq\_rrep\_sn (Rrep hops dip dsn oip sip)"}
  \texttt{shows "msg\_fresh \sigma (Rrep hops dip dsn oip sip)"
    (is "msg\_fresh \sigma ?msg")}
  \texttt{proof -}
  \texttt{let ?rt = "rt (\sigma sip)"
  from \texttt{assms} have "sip ≠ dip → dip∈kD(?rt) ∧ sqn ?rt dip = dsn ∧ the (flag ?rt dip) = val"
    by \texttt{simp}
  hence "sip ≠ dip → dip∈kD(?rt) ∧ nsqn ?rt dip ≥ dsn"
    by \texttt{clarsimp}
  with \texttt{assms} show "msg\_fresh \sigma ?msg"
    by \texttt{clarsimp}
  qed

lemma \texttt{rerr\_nsqn\_is\_fresh [simp]}:
  \texttt{fixes \sigma msg dests sip}
  \texttt{assumes "rerr\_invalid (rt (\sigma sip)) (Rerr dests sip)"
    shows "msg\_fresh \sigma (Rerr dests sip)"
    (is "msg\_fresh \sigma ?msg")}
  \texttt{qed}
proof -
let \(?rt = \text{rt (sip)}\)
from \(\text{assms have *: } (\forall \text{rip} \in \text{dom(dests)}. (\text{rip} \in \text{D}(\text{rt (sip)})) \wedge \text{the (dests rip) = sqn (rt (sip)) rip})\)
by clarsimp
have "(\(\forall \text{rip} \in \text{dom(dests)}. (\text{rip} \in \text{D}(\text{rt (sip})) \wedge \text{the (dests rip) - 1} \leq \text{nsqn (rt (sip)) rip})\)"
proof
fix \text{rip}
assume "\text{rip} \in \text{dom dests}"
with * have "\(\text{rip} \in \text{D}(\text{rt (sip)})\)" and "\text{the (dests rip) = sqn (rt (sip)) rip}" by auto
from this(2) have "\(\text{rip} \in \text{D}(\text{rt (sip)})\) ∧ \text{the (dests rip) - 1} \leq \text{nsqn (rt (sip)) rip}\)".
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 \(\text{assms}(2)\)
using \(\text{assms}(2)\)
proof (cases \text{m})
fix \text{hops dip dsn dsk oip osn sip handled}
assume [simp]: "m = Rreq hops dip dsn dsk oip osn sip handled"
and "msg_fresh \sigma m"
then have "\(\text{osn} \geq 1\)" and "\(\text{sip} = \text{oip} \lor (\text{oip} \in \text{D}(\text{rt (sip)}) \wedge \text{osn} \leq \text{nsqn (rt (sip)) oip} \wedge (\text{nsqn (rt (sip)) oip} = \text{osn} \rightarrow (\text{the (dhops (rt (sip)) oip) \leq hops} \lor \text{the (flag (rt (sip)) oip) = inv}))\)" by auto
from this(2) show "?thesis"
proof
assume "\text{sip} = \text{oip}" with \(\text{osn} \geq 1\) show "?thesis" by simp
next
assume "\(\text{oip} \in \text{D}(\text{rt (sip)}) \wedge \text{osn} \leq \text{nsqn (rt (sip)) oip} \wedge (\text{nsqn (rt (sip)) oip} = \text{osn} \rightarrow (\text{the (dhops (rt (sip)) oip) \leq hops} \lor \text{the (flag (rt (sip)) oip) = inv}))\)"
moreover from qinc have "quality_increases (\sigma sip) (\sigma' sip)"
ultimately have "\(\text{oip} \in \text{D}(\text{rt (sip)}) \wedge \text{osn} \leq \text{nsqn (rt (sip)) oip} \wedge (\text{nsqn (rt (sip)) oip} = \text{osn} \rightarrow (\text{the (dhops (rt (sip)) oip) \leq hops} \lor \text{the (flag (rt (sip)) oip) = inv}))\)"
using \(\text{osn} \geq 1\) by (rule quality_increases_rreq_rrep_props [rotated 2])
with \(\text{osn} \geq 1\) show "msg_fresh \sigma' m"
by (clarsimp)
qed
next
fix \text{hops dip dsn oip sip}
assume [simp]: "m = Rrep hops dip dsn oip sip" and "msg_fresh \sigma m"
then have "\(\text{dsn} \geq 1\)" and "\(\text{sip} = \text{dip} \lor (\text{dip} \in \text{D}(\text{rt (sip)}) \wedge \text{dsn} \leq \text{nsqn (rt (sip)) dip} \wedge (\text{nsqn (rt (sip)) dip} = \text{dsn} \rightarrow (\text{the (dhops (rt (sip)) dip) \leq hops} \lor \text{the (flag (rt (sip)) dip) = inv}))\)"
using \(\text{osn} \geq 1\) by (rule quality_increases_rreq_rrep_props [rotated 2])
with \(\text{osn} \geq 1\) show "msg_fresh \sigma' m"
by (clarsimp)
qed
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

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 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_sims)

lemma σAODV'_labels [simp]: "(σ, p) ∈ σAODV' → labels ΓAODV p = {PAodv−:0}"
  unfolding σAODV'_def by simp
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/OAWN_Convert" 
E_OAodv
begin

lemma other_quality_increases [elim]:
assumes "other quality_increases I σ σ'"
shows "∀ j. quality_increases (σ j) (σ' j)"
using assms by (rule, clarsimp) (metis quality_increases_refl)

lemma weaken_otherwith [elim]:
fixes m
assumes *: "otherwith P I (orecvmsg Q) σ σ' a"
and weakenP: "∀ m. P σ m ⇒ P' σ m"
and weakenQ: "∀ m. Q σ m ⇒ Q' σ m"
shows "otherwith P' I (orecvmsg Q') σ σ' a"
proof
fix j
assume "j ∉ I"
with * have "P (σ j) (σ' j)" by auto
thus "P' (σ j) (σ' j)" by (rule weakenP)
next
from * have "orecvmsg Q σ a" by auto
thus "orecvmsg Q' σ a"
by rule (erule weakenQ)
qed

lemma oreceived_msg_inv:
assumes other: "∀ σ σ' m. [ P σ m; other Q {i} σ σ' ] ⇒ P σ' m"
and local: "∀ σ m. P σ m ⇒ P (σ (i := σ i (msg := m))) m"
shows "opaodv i = (otherwith Q {i} (orecvmsg P), other Q {i}) onl Γ_AODV (λ (σ, l). l ∈ {PAodv-:1} −→ P σ (msg (σ i)))"
proof (inv_cterms, intro impI)
fix σ σ' l
assume "l = PAodv-:1 −→ P σ (msg (σ i))"
and "l = PAodv-:1"
and "other Q {i} σ σ'"
from this(1-2) have "P σ (msg (σ i))" ..
hence "P σ' (msg (σ i))" using 'other Q {i} σ σ'
by (rule other)
moreover from 'other Q {i} σ σ' have "σ' i = σ i" ..
ultimately show \( "P \sigma' (msg (\sigma' i))" \) by simp

next

fix \( \sigma \) \( \sigma' \) \( msg \)

assume "otherwith \( Q \{i\} (\text{orecvmsg } P) \sigma \sigma' (\text{receive } msg)"

and "\( \sigma' i = \sigma i[msg := msg] \)"

from (1) have "\( P \sigma msg \)"

and "\( \forall j. j \neq i \rightarrow Q (\sigma j) (\sigma' j) \)" by auto

from (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

(Equivalent to) Proposition 7.27

lemma local_quality_increases:

"paodv i \models (\text{recvmsg repreq_rrep_sn} \rightarrow \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{quality_increases } \xi (\xi')))"

proof (rule step_invariantI)

fix \( s \) \( a \) \( s' \)

assume sr: "\( s \in \text{reachable (paodv i) (recvmsg rreq_rrep_sn)} \)"

and tr: "\( (s, a, s') \in \text{trans (paodv i)} \)"

and rm: "\( \text{recvmsg rreq_rrep_sn } a \)"

from sr have srTT: "\( s \in \text{reachable (paodv i) TT} \)"

from route_tables_fresher sr tr rm

have "\( \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{ dip } \in kD \text{ (rt } \xi \text{). rt } \xi \subseteq \text{ dip } \text{ (rt } \xi') \text{ (s, a, s')} \)"

by (rule step_invariantD)

moreover from known_destinations_increase srTT tr TT_True

have "\( \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{ ip. sqn (rt } \xi \text{) ip } \leq \text{ sqn (rt } \xi') \text{ ip} \text{ (s, a, s')} \)"

by (rule step_invariantD)

moreover from sqns_increase srTT tr TT_True

have "\( \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \forall \text{ ip. sqn (rt } \xi \text{)} \text{ ip } \leq \text{ sqn (rt } \xi') \text{ ip} \text{ (s, a, s')} \)"

by (rule step_invariantD)

ultimately show "\( \text{onll } \Gamma_{AODV} (\lambda((\xi, _), _, (\xi', _)). \text{quality_increases } \xi (\xi') \text{ (s, a, s')} \)"

unfolding onll_def by auto

qed

lemmas olocal_quality_increases =

open_seq_step_invariant [OF local_quality_increases initiali_aodv oaodv_trans aodv_trans, simplified seqll_onll_swap]

lemma quality_increases:

"opaodv i \models (\text{otherwith quality_increases } \{i\} (\text{orecvmsg } (\lambda_. \text{rreq_rrep_sn})), other quality_increases (i) \rightarrow)

\text{onll } \Gamma_{AODV} (\lambda((\sigma, _), _, (\sigma', _)). \forall j. \text{quality_increases } (\sigma j) (\sigma' j))"

(is "" \models (\text{other } \{S\} \rightarrow _)"

) proof (rule onll_ostep_invariantI, simp)

fix \( \sigma \) \( \sigma' \) \( p \) \( p' \) \( l' \)

assume or: "\( (\sigma, p) \in \text{reachable (opaodv i) } ?S \) other quality_increases (\{i\})"

and 11: "\( l \in \text{labels } \Gamma_{AODV} p \)"

and "\( ?S \sigma \sigma' a \)"

and tr: "\( ((\sigma, p), a, (\sigma', p')) \in \text{oseqp_sos } \Gamma_{AODV} i \)"

and 11': "\( l' \in \text{labels } \Gamma_{AODV} p' \)"

from (1-3) have "\( \text{orecvmsg } (\lambda_. \text{rreq_rrep_sn}) \sigma a \)"

by (auto dest!: reachable_weakenE [where QS="act (recvmsg rreq_rrep_sn)"

and QU="other quality_increases {i}"]

otherwith_actionD)

with or have orw: "\( (\sigma, p) \in \text{reachable (opaodv i) (act (recvmsg rreq_rrep_sn))} \) other quality_increases {i}"

469
by - (erule reachable_weakenE, auto)
with tr 11 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 \( tS \sigma \sigma' a \) show \( \forall j. \text{quality_increases} \((\sigma j) (\sigma' j)\)"
by (auto dest!: otherwith_syncD)
qed

**lemma rreq_rrep_nsqn_fresh_any_step_invariant:**

\[\text{opaodv i} \models_A (\text{recvmsg rreq_rrep_sn}, \text{other A} \{i\} \rightarrow)\]

\[\text{oAODV} (\lambda(\sigma, ..), a, ..). \text{anycast} (\text{msg_fresh} \sigma a)\]

**proof (rule ostep_invariantI, simp del: act_simp)**

fix \( \sigma p a \sigma' p' \)
assume
- \( (\sigma, p) \in \text{oreachable (opaodv i) (act (recvmsg rreq_rrep_sn)) (other A \{i\})} \)
and 
- \( ((\sigma, p), a, (\sigma', p')) \in \text{oseqp_sos AODV i} \)

have \( \text{tr} ((\sigma, p), a, (\sigma', p')) \in \text{trans (opaodv i)} \)
by simp

have \( \text{anycast (rreq_rrep_fresh (rt (\sigma i))) a} \)
proof -

have \( \text{opaodv i} \models_A (\text{recvmsg rreq_rrep_sn}, \text{other A} \{i\} \rightarrow)\)

\[\text{oAODV} (\lambda((\xi, ..), a, ..). \text{anycast} (\text{rreq_rrep_fresh} (\text{rt} \xi) a))\]

by (rule ostep_invariant_weakenE [OF open_seq_step_invariant [OF rreq_rrep_fresh_any_step_invariant initiali_aodv, simplified seqll_onll_swap]) auto

hence \( \text{oAODV} (\text{seqll} i (\lambda((\xi, ..), a, ..). \text{anycast} (\text{rreq_rrep_fresh} (\text{rt} \xi) a)) (((\sigma, p), a, (\sigma', p')))) \)

using or tr recv by - (erule(4) ostep_invariantE)
thus \( ?\text{thesis} \)
using \( l \in \text{labels} \text{AODV} p \) and \( l' \in \text{labels} \text{AODV} p' \) by auto
qed

moreover have "anycast (rerr_invalid (rt (\sigma i))) a"
proof -

have "opaodv i \models_A (\text{recvmsg rreq_rrep_sn}, \text{other A} \{i\} \rightarrow)\)

\[\text{oAODV} (\text{seqll} i (\lambda((\xi, ..), a, ..). \text{anycast} (\text{rerr_invalid} (\text{rt} \xi) a)))\]

by (rule ostep_invariant_weakenE [OF open_seq_step_invariant [OF rerr_invalid_any_step_invariant initiali_aodv, simplified seqll_onll_swap]) auto

hence "\text{oAODV} (\text{seqll} i (\lambda((\xi, ..), a, ..). \text{anycast} (\text{rerr_invalid} (\text{rt} \xi) a)) (((\sigma, p), a, (\sigma', p')))) \)

using or tr recv by - (erule(4) ostep_invariantE)
thus \( ?\text{thesis} \)
using \( l \in \text{labels} \text{AODV} p \) and \( l' \in \text{labels} \text{AODV} p' \) by auto
qed

moreover have "anycast \( \lambda m. \text{not_Pkt} m \rightarrow \text{msg_sender} m = i \) a"
proof -

from or tr recv

have "\text{oAODV} (\text{seqll} i (\lambda((\xi, ..), a, ..). \text{anycast} (\text{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 aodv_trans, simplified seqll_onll_swap])

thus \( ?\text{thesis} \)
using \( l \in \text{labels} \text{AODV} p \) and \( l' \in \text{labels} \text{AODV} p' \) by auto
qed

moreover have "anycast (\lambda m. \text{not_Pkt} m \rightarrow \text{msg_sender} m = i) a"
proof -

have "opaodv i \models_A (\text{recvmsg rreq_rrep_sn}, \text{other A} \{i\} \rightarrow)\)

\[\text{oAODV} (\text{seqll} i (\lambda((\xi, ..), a, ..). \text{anycast} (\lambda m. \text{not_Pkt} m \rightarrow \text{msg_sender} m = i) a)))\]
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 Γ_AODV (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 Γ_AODV (λ(ξ, 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
qed

lemma oquality_increases_nsqn_fresh:
"opaodv i |=( otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} →)
onll Γ_AODV (λ((σ, l). l ∈ {PAodv-:4, PAodv-:5} ∪ {PRreq-:0, PRreq-:2} −→ 1 ≤ osn ξ))"
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} →)
onll Γ_AODV (seql 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: seql_onl_swap)

lemma rreq_sip:
"opaodv i |=( otherwith quality_increases {i} (orecvmsg msg_fresh),
other quality_increases {i} →)
onll Γ_AODV (λ(ξ, l). l ∈ {PAodv-:4, PAodv-:5, PRreq-:0, PRreq-:2} ∧ sip (σ i) ≠ oip (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 oadov_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 impl)

fix σ σ' p l
assume "("σ, p) ∈ orreachable (opaodv i) ?S ?U"
and "l ∈ labels Γ_AODV p"
and pre:

"(l = PAodv-:4 ∨ l = PAodv-: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 = PAodv-:4 ∨ l = PAodv-: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: simp add: seqlsimp)
moreover from ⟨σ, p⟩ ∈ orreachable (opaodv i) ?S ?U | l ∈ labels Γ_AODV p and hyp
have "1 ≤ osn (σ i)"
by (auto dest!: onl_oinvrnt_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
next
assume "sip (σ i) = i" thus ?thesis
using σ' i = σ i hyp and pre by auto
qed

(auto elim!: quality_increases_rreq_rrep_props')

lemma odsn_rrep:
"opaodv i ⊢ (otherwith quality_increases {i} (orecmsg msg_fresh),
other_quality_increases {i} →)
\(\Gamma_{AODV} \seq i (\lambda x (1, 1). 1 \in \{PAodv-:6, PAodv-:7\} \cup \{PRrep-n1:1, \_\} \rightarrow 1 \leq \dsn x))\)
by (rule oinvrnt_weakenE [OF open_seq_invariant [OF dsn_rrep initiai_aodv]])
(auto simp: seql_onl_swap)

lemma rrep_sip:
"opaodv i ⊢ (otherwith quality_increases {i} (orecmsg msg_fresh),
other_quality_increases {i} →)
\(\Gamma_{AODV} \lambda (\sigma, 1).
\(1 \in \{PAodv-:6, PAodv-:7, PRrep-0, PRrep-1\} \land \sip (\sigma i) \neq \dip (\sigma i)
\rightarrow \dip (\sigma i) \in kd (rt (\sigma (\sip (\sigma i))))
\land \nsqn (rt (\sigma (\sip (\sigma i)))) (\dip (\sigma i)) \geq \dsn (\sigma i)
\land (\nsqn (rt (\sigma (\sip (\sigma i)))) (\dip (\sigma i)) = \dsn (\sigma i))
\rightarrow (\hops (\sigma i) \geq the (dhops (rt (\sigma (\sip (\sigma i)))) (\dip (\sigma i))))
\lor the (flag (rt (\sigma (\sip (\sigma i)))) (\dip (\sigma i))) = inv))\)"
(is "$_ {\sim} (\?S, \?U \rightarrow \_) \)

472
lemma rerr_sip:
"opaodv i |= (otherwith quality_increases {i} (orecvmsg msg_fresh),
  other_quality_increases {i} →)
  onl Γ_AODV (λ(σ, i).
    l ∈ {PAodv→:8, PAodv→:9, PRerr→:0, PRerr→:1} → (∀ ripc∈dom(dests (σ i)).
      ripc∈kd (rt (σ (sip i))) ∧ the (dests (σ i) ripc) - 1 ≤ nsqn (rt (σ (sip i))) ripc))"
(is "_ |= (?S, ?U →) _")

proof -
  { fix dests rip sip ran 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 ran"
    from this(3) have "rip∈dom dests" by auto

fix σ σ' p l
assume "(σ, p) ∈ oreachable (opaodv i) ?S ?U"
and "1 ∈ labels Γ_AODV p"
and pre:
"(1 = PAodv→:6 ∨ 1 = PAodv→:7 ∨ 1 = PRrep→:0 ∨ 1 = PRrep→:1) ∧ sip (σ i) ≠ dip (σ i)
  → dip (σ i) ∈ kd (rt (σ (sip (σ i))))
 ∧ (nsqn (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"
  and "other_quality_increases {i} σ σ'"
  and hyp: "(1=PAodv→:6 ∨ 1=PAodv→:7 ∨ 1=PRrep→:0 ∨ 1=PRrep→:1) ∧ sip (σ' i) ≠ dip (σ' i)"
  (is "?labels ∧ sip (σ' i) ≠ dip (σ' i)"
  from(4) have "σ' i = σ i" ..
  with hyp have hyp'": "?labels ∧ sip (σ i) ≠ dip (σ i)" by simp
  show "dip (σ' i) ∈ kd (rt (σ' (sip (σ i))))
  ∧ (nsqn (σ i) (sig (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"
  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: (sig (σ i)) ≠ i)
    moreover from "(σ, p) ∈ oreachable (opaodv i) ?S ?U: 1 ∈ labels Γ_AODV p" and hyp
    have "1 ≤ dsn (σ' i)"
      by (auto dest!: onl_oivarient_weakenD [OF odsn_rrep]
          simp add: seqlsimp (sig i) = i)
    moreover from "sip (σ i) ≠ i" hyp' and pre
    have "dip (σ' i) ∈ kd (rt (σ' (sip (σ i))))
  ∧ (nsqn (σ i) (sig (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: (sig i) = i)
      ultimately show ?thesis
          by (rule quality_increases_rreq_rrep_props)
  next
    assume "sip (σ i) = i" thus ?thesis
    using (sig i) = i hyp and pre by auto
  qed
  (auto simp add: seqlsimp elim!: quality_increases_rreq_rrep_props')

proof (inv_cterms inv add: oseq_step_invariant_sterms [OF quality_increases nsqn_fresh aodv_wf]
  aodv_trans)
  onl_oivarient_sterms [OF aodv_wf oreceived_rreq_rrep nsqn_fresh_inv]
  onl_oivarient_sterms [OF aodv_wf odsn_rrep]
  simp del: One_nat_def, rule impI)

with \( \text{dests rip} = \text{Some rsn} \) have "rip \( \in kD(rt (\sigma \text{ sip})) \) "
and "rsn - 1 \( \leq \text{nsqn (rt (\sigma \text{ sip})) rip} \) "

by (auto dest!: bspec)

from qinc have "quality_increases (\sigma \text{ sip}) (\sigma \text{' sip})" ..
have "rip \( \in kD(rt (\sigma \text{' sip})) \wedge rsn - 1 \leq \text{nsqn (rt (\sigma \text{' sip})) rip} \) "

proof

from \( \langle \text{rip} \in kD(rt (\sigma \text{ sip})) \rangle \) and \( \langle \text{quality_increases (\sigma \text{ sip}) (\sigma \text{' sip})} \rangle \) show "rip \( \in kD(rt (\sigma \text{' sip})) \) " ..

next

from \( \langle \text{rip} \in kD(rt (\sigma \text{ sip})) \rangle \) and \( \langle \text{quality_increases (\sigma \text{ sip}) (\sigma \text{' sip})} \rangle \) have "nsqn (rt (\sigma \text{ sip})) rip \( \leq \text{nsqn (rt (\sigma \text{' sip})) rip} \) " ..

with \( \langle \text{rsn - 1 \leq \text{nsqn (rt (\sigma \text{ sip})) rip} \rangle \) show "rsn - 1 \( \leq \text{nsqn (rt (\sigma \text{' sip})) rip} \) "

by (rule le_trans)

qed

note partial = this

show ?thesis

by (inv_cterms inv add: oseq_step_invariant_sterms [OF equality_increases_nsqn_fresh aodv_wf aodv_trans]
onl_oinvariant_sterms [OF aodv_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: "paodv i =
  onl \( \Gamma_{AODV} (\lambda (\xi, l). (l = \text{PRerr-1} -\rightarrow (\forall ip \in \text{dom(dests } \xi). ip \in \text{vD(rt } \xi)
  \wedge \text{the (nhop (rt } \xi) ip) = \text{sip } \xi
  \wedge \text{sqn (rt } \xi) ip < \text{the (dests } \xi \text{ 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_aodv oaodv_trans aodv_trans,
simplified seql_onl_swap,
THEN oinvariant_anyact]

lemmas oprerr_guard =
open_seq_invariant [OF prerr_guard initiali_aodv oaodv_trans aodv_trans,
simplified seql_onl_swap,
THEN oinvariant_anyact]

Proposition 7.28

lemma seq_compare_next_hop':
"opaodv i = (otherwith quality_increases \{i\} (orecmsg msg_fresh),
other_quality_increases \{i\} \rightarrow onl \( \Gamma_{AODV} (\lambda(\sigma, _). \forall \text{dip. let nhip = the (nhop (rt } \sigma \text{i)} dip)
  \rightarrow \text{dip} \in kD(rt (\sigma \text{nhip})) \wedge \text{nsqn (rt } \sigma \text{i)} dip \leq \text{nsqn (rt (\sigma \text{nhip)} dip})
" (is "_ = (?S, ?U \rightarrow ) _")

proof -

{ fix nhip and \( \sigma \text{ and } \sigma' :: "ip \Rightarrow state" 
  assume pre: "\forall \text{dip} \in kD(rt (\sigma \text{i)}). \text{nhop dip} \neq \text{dip} \rightarrow \text{dip} \in kD(rt (\sigma \text{(nhop dip)}) \wedge \text{nsqn (rt } \sigma \text{i)} dip \leq \text{nsqn (rt (\sigma \text{(nhop dip)}) dip})"
  and qinc: "\forall j. quality_increases (\sigma \text{j}) (\sigma \text{' j})"
  have "\forall \text{dip} \in kD(rt (\sigma \text{i)}). \text{nhop dip} \neq \text{dip} \rightarrow \text{dip} \in kD(rt (\sigma \text{' (nhop dip)}) \wedge \text{nsqn (rt } \sigma \text{i)} dip \leq \text{nsqn (rt (\sigma \text{' (nhop dip)}) dip})"
  proof (intro ballI impI)
    fix dip
    assume "\text{dip} \in kD(rt (\sigma \text{i})"
    and "\text{nhop dip} \neq \text{dip}"
    with pre have "\text{dip} \in kD(rt (\sigma \text{(nhop dip)})"
    and "\text{nsqn (rt } \sigma \text{i)} dip \leq \text{nsqn (rt (\sigma \text{(nhop dip)}) dip}"

474
by auto
from qinc have qinc_nhph: "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_nhph
  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 nhph 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 \sigma \sigma' :: "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"
(prove by (rule basic, auto)
thus "∀ dip∈kD(rt (σ i)). ?P dip")
qed

note nhop_update_sip = this

\{ fix \sigma 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"
(prove 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")
(prove (rule, split update_rt_split_asm)
assume "(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

476
\{ note update1 = this

\{ fix \sigma \sigma' oip sip osn hops
  assume pre: "\forall \text{dip} \in kD (rt (\sigma i)). the (nhop (rt (\sigma i)) dip) \neq dip
  \rightarrow dip \in kD (rt (\sigma (\text{the (nhop (rt (\sigma i)) dip))))
  \land \text{nsqn (rt (\sigma i)) dip} \leq \text{nsqn (rt (\sigma (\text{the (nhop (rt (\sigma i)) dip)))) dip}"
  and qinc: "\forall j. quality_increases (\sigma j) (\sigma', j)"
  and *: "sip \neq oip \rightarrow dip \in kD (rt (\sigma sip))
  \land osn \leq \text{nsqn (rt (\sigma sip)) oip}
  \land (\text{nsqn (rt (\sigma sip)) oip = osn}
  \rightarrow \text{the (dhops (rt (\sigma sip)) oip) \leq hops}
  \lor \text{the (flag (rt (\sigma sip)) oip) = inv)}"

  from pre and qinc
  have pre': "\forall \text{dip} \in kD (rt (\sigma i)). the (nhop (rt (\sigma i)) dip) \neq dip
  \rightarrow dip \in kD (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip))))
  \land \text{nsqn (rt (\sigma i)) dip} \leq \text{nsqn (rt (\sigma' (\text{the (nhop (rt (\sigma i)) dip)))) dip}"

  by (rule basic)
  have *: "\forall \text{dip} \in kD (rt (\sigma i)).
          \text{the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip) \neq dip}
          \rightarrow dip \in kD (rt (\sigma' (\text{the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip))))
          \land \text{nsqn (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip}
          \leq \text{nsqn (rt (\sigma' (\text{the (nhop (update (rt (\sigma i)) oip (osn, kno, val, Suc hops, sip)) dip)))) dip}"

          (is "\forall \text{dip} \in kD (rt (\sigma i)). _ \rightarrow ?\text{dip}_\text{in}_kD \text{ dip} \land ?\text{nsqn}_\text{le}_\text{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 "?\text{dip}_\text{in}_kD \text{ dip} \land ?\text{nsqn}_\text{le}_\text{nsqn} \text{ 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 "?\text{dip}_\text{in}_kD \text{ dip} \land ?\text{nsqn}_\text{le}_\text{nsqn} \text{ 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 "?\text{dip}_\text{in}_kD \text{ dip}"
      by simp (metis kD_quality_increases)
    moreover from (dip = oip) rtnot qinc (dip \in kD (rt (\sigma i))) notdip *
      have "?\text{nsqn}_\text{le}_\text{nsqn} \text{ dip}" by simp (metis kD_quality_increases_trans)
    ultimately show ?thesis ..
  qed
  qed
\} note update2 = this

have "opaodv i \models (?S, \emptyset \rightarrow) \text{onl} \Gamma_{ADV} (\lambda (\sigma, _)).
\forall \text{dip} \in kD (rt (\sigma i)). the (nhop (rt (\sigma i)) dip) \neq dip
\rightarrow dip \in kD (rt (\sigma (\text{the (nhop (rt (\sigma i)) dip))))
\land \text{nsqn (rt (\sigma i)) dip} \leq \text{nsqn (rt (\sigma (\text{the (nhop (rt (\sigma i)) dip)))) dip})"
by (inv_cterms inv add: oseq_step_invariant_sterms [OF quality_increases_nsqn_fresh aodv_wf oaoadv_trans]
  oshr_loinvariant_sterms [OF aodv_wf oadvtodest_v0_inc_sqn]
  oshr_loinvariant_sterms [OF aodv_wf oprerr_guard]
  oshr_loinvariant_sterms [OF aodv_wf rreq_sip]
  oshr_loinvariant_sterms [OF aodv_wf rrep_sip]"
thus \(?thesis\) unfolding \(\text{Let_def}\) by auto
qed

Proposition 7.30
lemmas \(\text{oKD\_unk\_or\_atleast\_one} = \)
\(\text{open\_seq\_invariant} \ [\text{OF kD\_unk\_or\_atleast\_one\ initiali\_aodv}, \)
\(\text{simplified seql\_onl\_swap}]\)
lemmas \(\text{ozero\_seq\_unk\_hops\_one} = \)
\(\text{open\_seq\_invariant} \ [\text{OF zero\_seq\_unk\_hops\_one\ initiali\_aodv}, \)
\(\text{simplified seql\_onl\_swap}]\)

lemma \(\text{oreachable\_fresh\_oKD\_unk\_or\_atleast\_one}\):
fixes \(dip\)
assumes \("(\sigma, p) \in \text{oreachable (opaodv i})\)
\((\text{otherwith (op=} \{i\} (\text{orecvmsg (}\lambda\sigma \ m. \ msg\_fresh \sigma \ m \)
\wedge \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} \lor 1 \leq \pi_2(\text{the (}\text{rt (}\sigma \ i) \text{ dip}))")"
(is \("\text{?P dip}\)"
proof -
  have \("\exists l. \ l\in\text{labels} \ \Gamma_{AODV} \ p\)" by (metis aodv_ex_label)
with asms(1) have \("\text{?dip}\in\text{kD}(\text{rt (}\sigma \ i))\). \text{?P dip}\)"
by - (drule oinvariant\_weakenD [OF oKD\_unk\_or\_atleast\_one [OF oao\_trans aodv\_trans]],
auto dest!: oinvariant\_actionD oon\_simp: seql\_simp)
with \(\langle \text{dip}\in\text{kD}(\text{rt (}\sigma \ i))\rangle\) show \(?thesis\) by simp
qed

lemma \(\text{oreachable\_fresh\_ozero\_seq\_unk\_hops\_one}\):
fixes \(dip\)
assumes \("(\sigma, p) \in \text{oreachable (opaodv i})\)
\((\text{otherwith (op=} \{i\} (\text{orecvmsg (}\lambda\sigma \ m. \ msg\_fresh \sigma \ m \)
\wedge \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 \rightarrow \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}) = dip})")"
(is \("\text{?P dip}\)"
proof -
  have \("\exists l. \ l\in\text{labels} \ \Gamma_{AODV} \ p\)" by (metis aodv_ex_label)
with asms(1) have \("\text{?dip}\in\text{kD}(\text{rt (}\sigma \ i))\). \text{?P dip}\)"
by - (drule oinvariant\_weakenD [OF ozero\_seq\_unk\_hops\_one [OF oao\_trans aodv\_trans]],
auto dest!: oon\_otherwith\_actionD oon\_simp: seql\_simp)
with \(\langle \text{dip}\in\text{kD}(\text{rt (}\sigma \ i))\rangle\) show \(?thesis\) by simp
qed

lemma \(\text{seq\_nhop\_quality\_increases}'\):
shows \("\text{opaodv i \models (otherwith (op=} \{i\})}
\((\text{orecvmsg (}\lambda\sigma \ m. \ msg\_fresh \sigma \ m \wedge \msg\_zhops \ m)))\),
\(\text{other quality\_increases \{i\} \rightarrow}
\(\text{onl} \ \Gamma_{AODV} \ (\lambda(\sigma, \_). \ \forall \text{dip}. \ \text{let nhip = the (}\text{nhop (}\text{rt (}\sigma \ i) \text{ dip}) \text{ in dip} \in \text{vD (}\text{rt (}\sigma \ i) \text{)} \cap \text{vD (}\text{rt (}\sigma \ nhhip) \wedge \text{nhhip} \neq \text{dip})})\)"
478
is \( _\rightarrow \) (?S i, _ \rightarrow _\) _

proof -

have weaken:
\[ \forall P Q R. P \models (otherwith quality_increases I (orecvmsg Q), other quality Increases I \rightarrow) P \Rightarrow P \models (otherwith (op=) I (orecvmsg (λσ m. Q σ m \land R σ m)), other quality Increases I \rightarrow) P \]
by auto

\{ fix i a and σ σ' :: "i op = state"
assume a1: "∀ dip. dip∈vD(rt (σ i))
\land dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))
\land (the (nhop (rt (σ i)) dip)) \neq dip
\rightarrow rt (σ i) \sqsubseteq dip rt (σ (the (nhop (rt (σ i)) dip)))"
and ov: "?S i σ σ' a"

have "∀ dip. dip∈vD(rt (σ i))
\land dip∈vD (rt (σ' (the (nhop (rt (σ i)) dip))))
\land (the (nhop (rt (σ i)) dip)) \neq dip
\rightarrow rt (σ i) \sqsubseteq 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)) \neq dip"

from ov have "∀ j. j \neq i \rightarrow j = σ' j" by auto

show "rt (σ i) \sqsubseteq 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)) \neq 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) \sqsubseteq dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp

with (the (nhop (rt (σ i)) dip)) = i have "rt (σ i) \sqsubseteq dip rt (σ (the (nhop (rt (σ i)) dip)))" by simp

hence False by simp

thus ?thesis ..

next

assume "(the (nhop (rt (σ i)) dip)) = i"

with \( \forall j. j \neq i \rightarrow j = σ' j \)

have *. "σ (the (nhop (rt (σ i)) dip)) = σ' (the (nhop (rt (σ i)) dip))" by simp

with \( dip ∈ vD (rt (σ i)) \) have "dip ∈ vD(rt (σ (the (nhop (rt (σ i)) dip))))" by simp

with a1 a2 a4 have "rt (σ i) \sqsubseteq 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))
\land dip∈vD(rt (σ (the (nhop (rt (σ i)) dip))))
\land (the (nhop (rt (σ i)) dip)) \neq dip
\rightarrow rt (σ i) \sqsubseteq dip rt (σ (the (nhop (rt (σ i)) dip)))"

and ov: "?S i σ σ' a"

have "∀ dip. dip∈vD(update rt (σ i) sip (0, unk, val, Suc 0, sip))
\land dip∈vD(rt (σ' (the (nhop (update rt (σ i) sip (0, unk, val, Suc 0, sip)) dip))))
\land (the (nhop (update rt (σ i) sip (0, unk, val, Suc 0, sip)) dip)) \neq dip
\rightarrow update (rt (σ i) sip (0, unk, val, Suc 0, sip))
\sqsubseteq 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) \neq dip"

show "update (rt (σ i) sip (0, unk, val, Suc 0, sip))
\sqsubseteq dip rt (σ' (the (nhop (update rt (σ i) sip (0, unk, val, Suc 0, sip)) dip)))"

proof (cases "dip = sip")

479
assume "dip = sip"
with \(\langle \text{the} (\text{nhop} (\text{update} (\text{rt} (\sigma i)) \text{ sip} (0, \text{unk}, \text{val}, \text{Suc} 0, \text{sip})) \text{ dip} \rangle \neq \text{ dip}\)" have False by simp 
thus \(?thesis \).
next
assume [simp]: "dip \neq \text{ sip}"
from a2 have "\(\text{ dip} \in \text{ vD} (\text{ rt} (\sigma i)) \lor \text{ dip} = \text{ sip}\)"
by (rule \text{ vD_update_val})
with \(\text{ dip} \neq \text{ sip}\) have "\(\text{ dip} \in \text{ vD} (\text{ rt} (\sigma i))\)" by simp
moreover from a3 have "\(\text{ dip} \in \text{ vD} (\text{ rt} (\sigma' (\text{ the} (\text{ nhop} (\text{ rt} (\sigma i)) \text{ dip})))\)" by simp
moreover from a4 have "the (nhop (rt (\sigma i)) \text{ dip} \neq \text{ dip})" by simp
ultimately have "\(\text{ rt} (\sigma i) \sqsubseteq \text{ dip} \text{ rt} (\sigma' (\text{ the} (\text{ nhop} (\text{ rt} (\sigma i)) \text{ dip})))\)"
using a1 ow by - (drule(1) basic, simp)
with \(\text{ dip} \neq \text{ sip}\) show \(?thesis\)
by - (erule \text{ rt_strictly_fresher_update_other}, simp)
qed
}

} note update_0_unk = this

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

} note invalidate = this

\{
fix \(\sigma\) a \(\sigma'\) dip oip osn sip hops i
assume pre: "\(\forall \text{ dip}. \text{ dip} \in \text{ vD} (\text{ rt} (\sigma i))\)
\land \text{ dip} \in \text{ vD} (\text{ rt} (\sigma (\text{ the} (\text{ nhop} (\text{ rt} (\sigma i)) \text{ dip})))\)
\land \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 \text{ osn}\"
and a6: "\(\text{ sip} \neq \text{ oip} \rightarrow \text{ oip} \in \text{ kD} (\text{ rt} (\sigma \text{ sip}))\)"
\land \(\text{ osn} \leq \text{ nsqn} (\text{ rt} (\sigma \text{ sip})) \text{ oip}\)
\land \(\text{ nsqn} (\text{ rt} (\sigma \text{ sip})) \text{ oip} = \text{ osn}\)
\rightarrow \text{ the} (\text{ dhops} (\text{ rt} (\sigma \text{ sip})) \text{ oip}) \leq \text{ hops}\n\lor \text{ the} (\text{ flag} (\text{ rt} (\sigma \text{ sip})) \text{ oip}) = \text{ inv}"
and after: "\(\sigma' i = \sigma i [\text{rt} := \text{ update} (\text{ rt} (\sigma i)) \text{ oip} (\text{ osn}, \text{ kno}, \text{ val}, \text{ Suc hops}, \text{ sip})]\)"
have "\(\forall \text{ dip}. \text{ dip} \in \text{ vD} (\text{ update} (\text{ rt} (\sigma i)) \text{ oip} (\text{ osn}, \text{ kno}, \text{ val}, \text{ Suc hops}, \text{ sip}))\)
\land \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}))))\)
\land \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}\n\rightarrow \text{ update} (\text{ rt} (\sigma i)) \text{ oip} (\text{ osn}, \text{ kno}, \text{ val}, \text{ Suc hops}, \text{ sip})\n\sqsubseteq \text{ dip}
\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})))"
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))) 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))) 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 ∈ vD (rt (σ sip))⟩
      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
      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 ..
next
  assume "oip ∈ iD (rt (σ sip))"
  with ⟨the (flag (rt (σ' sip)) oip) = val⟩ 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 simp
  with ⟨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)
  with ⟨osn ≤ nsqn (rt (σ sip)) oip⟩
  have "osn < nsqn (rt (σ' sip)) oip" by simp
  thus ?thesis ..
qed
thus \( \text{thesis} \)
proof
  assume osnlt: "osn < nsqn (rt (\(\sigma'\) sip)) oip"
  from \( \langle \text{dip} \in kD (rt (\(\sigma'\) i)) \rangle \) and \( \langle \text{dip} = oip \rangle \) have "\(\text{dip} \in kD (\langle rt \rangle (\text{rt1}))\)" by simp
  moreover from a3 have "\( \text{dip} \in kD(\text{rt2 dip}) \)" by simp
  moreover have "\(\text{nsqn } \text{rt1 dip} < \text{nsqn } \text{rt2 dip} \)"
    proof
      have "\(\text{nsqn } \text{rt1 oip} = osn\)"
        by (simp add: osnlt)
      also have "... < nsqn (rt (\(\sigma'\) sip)) oip" using osnlt.
      also have "... = nsqn (\(\text{rt2 oip}\) oip)" by (simp add: change)
      finally show \( \text{thesis} \)
        using \( \langle \text{dip} = oip \rangle \) by simp
    qed
  ultimately show \( \text{thesis} \)
    by (rule rt_strictly_fresher_ltI)
next
  assume osneq: "osn = nsqn (rt (\(\sigma'\) sip)) oip \& \text{the } (\text{dhops } (rt (\(\sigma'\) sip)) oip) \leq \text{hops}"
  have "\(\text{oip} \in kD (\langle \text{rt1} \rangle)\)" by simp
  moreover from a3 \( \langle \text{dip} = oip \rangle \) have "\(\text{oip} \in kD(\langle \text{rt2 oip}\rangle)\)" by simp
  more over have "\(\text{nsqn } \text{rt1 oip} = \text{nsqn } (\text{rt2 oip}) oip\)"
    proof
      from osneq have "osn = nsqn (rt (\(\sigma'\) sip)) oip" ..
      also have "osn = nsqn \(\text{rt1 oip}\)"
        by (simp add: osneq)
      also have "nsqn (rt (\(\sigma'\) sip)) oip = nsqn (\(\text{rt2 oip}\) oip) oip"
        by (simp add: change)
      finally show \( \text{thesis} \).
    qed
  ultimately have "\(\pi_5(\text{the } (\text{rt2 oip}) oip) < \pi_5(\text{the } (\text{rt1 oip}))\)"
    proof
      from osneq have "the (dhops (rt (\(\sigma'\) sip)) oip) \leq \text{hops}" ..
      moreover from \( \text{oip} \in vD (rt (\(\sigma'\) sip)) \) have "\(\text{oip} \in kD(rt (\(\sigma'\) sip))\)" by auto
      ultimately have "\(\pi_5(\text{(the } (\text{rt (\(\sigma'\) sip}) oip)) \leq \text{hops}\)"
        by (auto simp add: proj5_eq_dhops)
      also from change after have "\(\text{hops} < \pi_5(\text{(rt (\(\sigma'\) i) oip)})\)"
        by (simp add: proj5_eq_dhops) (metis dhops_update_changed_lessI)
      finally have "\(\pi_5(\text{(the } (\text{rt (\(\sigma'\) sip} oip))) < \pi_5(\text{(the } (\text{rt (\(\sigma'\) i}) oip))\)\) .
      with change after show \( \text{thesis} \) by simp
    qed
  ultimately have "\(\text{rt1} \sqsubseteq_{\text{oip}} \text{rt2 oip}\)"
    by (rule rt_strictly_fresher_eqI)
  with \( \langle \text{dip} = oip \rangle \) show \( \text{thesis} \) by simp
qed
qed
}

note rreq_rrep_update = this

have "\(\text{opaodv i} \models (\text{otherwith } (\text{op}=) \{i\}) (\text{orecvmsg } (\lambda \sigma \eta. \text{msg_fresh } \sigma \eta \land \text{msg_zhops } \eta))\),
  \text{other_quality_increases } \{i\} \rightarrow"
  onl \( \Gamma_{\text{AODV}} \)
    (\(\lambda(\sigma, _). \forall \text{dip} \in vD (rt (\sigma i)) \cap vD (rt (\sigma (\text{the } (\text{nhop } (rt (\sigma i)) \text{ dip})))) \land \text{the } (\text{nhop } (rt (\sigma i)) \text{ dip}) \neq \text{dip}
        \rightarrow rt (\sigma i) \sqsubseteq_{\text{dip}} rt (\sigma (\text{the } (\text{nhop } (rt (\sigma i)) \text{ dip}))))\)"
  proof (inv_cterms inv add: onl_oinvariant_sterms [OF aodv_wf rreq_sip [THEN weaken]]
    onl_oinvariant_sterms [OF aodv_wf rrep_sip [THEN weaken]]
    onl_oinvariant_sterms [OF aodv_wf rerr_sip [THEN weaken]]

483
fix σ σ' p l
assume or: "(σ, p) ∈ orachable (opaodv i) (?S i) (other quality_increases {i})"
and "other quality_increases {i} σ σ'"
and ll: "l ∈ labels Γ_AODV p"
and pre: "∀ dip. dip ∈ vD(rt (σ i)) ∧ dip ≠ nhip → rt (σ i) ⊏ dip rt (σ (the (nhop (rt (σ i)) dip)))"  
from this(1-2) have or': "(σ', p) ∈ orachable (opaodv i) (?S i) (other quality_increases {i})"  
by (rule oreachable_other')  
from or and ll have next_hop: "∀ dip. dip ∈ kD(rt (σ i)) ∧ dip ≠ nhip → dip ∈ kD(rt (σ nhip)) ∧ sqn (rt (σ i)) dip = sqn (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 oaodv_trans aodv_trans]] otherwith_actionD simp: seqsimp)

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 nsqns_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 ⟨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∈iD(rt (σ' ?nhip))⟩
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)

thus ?thesis unfolding Let_def .

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:
defines "opaodv i |= (otherwith (op=) {i})
   (orecmsg (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

5.10 Routing graphs and loop freedom

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

Given the state of a network σ, 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. r dip = Some (dsn, dsk, val, hops, ip')))"
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.

486
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 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: "[\[ dip ∈ vD(rt (σ nhip)); nhip ≠ dip \] \implies 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

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 σ s a σ' s'
  assumes "(((σ, s), a, (σ', s')) ∈ oparp_sos i (oseqp_sos Γ_AODV i) (seqp_sos Γ_QMSG))" 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 \langle\langle i qmsg \mid= (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 \langle\langle i qmsg \mid= A (otherwith (op=) \{i\} (orecvmsg (λ_. rreq_rrep_sn)) σ σ', other (λ_ _. True) \rightarrow)
    globala (λ(σ, a, σ'). quality_increases (σ i) (σ' i))" proof (rule ostep_invariant_weakenE [OF olocal_quality_increases], simp_all)
    fix t :: "(((nat ⇒ state) × (state, msg, pseqp, pseqp label) seqp), msg seq_action) transition"
    assume "onll Γ_AODV (λ((σ, _), _, (σ', _)). 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_nsqn_fresh_any_step:
  shows "opaodv i \langle\langle i qmsg \mid= A (otherwith (op=) \{i\} (orecvmsg (λσ m. msg_fresh σ m ∧ msg_zhops m)),
    other quality_increases \{i\} →)
    globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)" proof
  have "opaodv i \mid= A (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

lemma par_rreq_rrep_sn_quality_increases:
  "opaodv i \langle\langle i qmsg \mid= A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ), other (λ_ _. True) \rightarrow)
    globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))" proof
  have "opaodv i \mid= A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ), other (λ_ _. True) \rightarrow)
    globala (λ(σ, _, σ'). quality_increases (σ i) (σ' i))"
  by (rule ostep_invariant_weakenE [OF olocal_quality_increases]) (auto dest!: onllD seqllD elim!: aodv_ex_labelE)
  hence "opaodv i \langle\langle i qmsg \mid= A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ), other (λ_ _. True) \rightarrow)
    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 "opaodv i \langle\langle i qmsg \mid= A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_ _. True) \rightarrow)
    globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)" proof
  have "opaodv i \mid= A (λσ _. orecvmsg (λ_. rreq_rrep_sn)) σ, other (λ_ _. True) \rightarrow)
    globala (λ(σ, a, σ'). anycast (msg_fresh σ) a)"
  proof (rule ostep_invariant_weakenE [OF qreq_rrep_nsqn_fresh_any_step_invariant])
    fix t
  qed

488
assume "onll Γ_AODV (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 aodv_ex_label)
qed

hence "opaodv i ⟨⟨i qmsg ≌ A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →)
globala (λ(λ(σ, a, _). anycast (msg_fresh σ) a)) t" by (rule lift_step_into_qmsg_statelessassm) simp_all
thus ?thesis by rule auto
qed

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

thus "opaodv i ⟨⟨i qmsg ≌ A (λσ _. orecvmsg (λ_. rreq_rrep_sn) σ, other (λ_. True) {i} →)
globala (λ(λ(σ, a, _). anycast msg_zhops a)) t"
proof (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 "((σ, NodeS i P R), a, (σ', NodeS i' P' R')) ∈ onode_sos (oparp_sos i (oseqp_sos Γ_AODV i) (seqp_sos Γ_QMSG))"
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_nhops_quality_increases:
shows "i : opaodv i ⟨⟨i qmsg : R ᵉ⟩. (oparp_sos i (oseqp_sos Γ_AODV i) (seqp_sos Γ_QMSG))"
proof (rule node_lift [OF par_nhops_quality_increases]) auto

lemma node_quality_increases:
shows "i : opaodv i ⟨⟨i qmsg : R ᵉ⟩. (oparp_sos i (oseqp_sos Γ_AODV i) (seqp_sos Γ_QMSG))"
by (rule node_lift_step_statelessassm [OF par_rreq_rrep_sn_quality_increases]) simp

lemma node_rreq_rrep_nsqn_fresh_any_step:
shows "i : opaodv i ⟨⟨i qmsg : R ᵉ⟩. (oparp_sos i (oseqp_sos Γ_AODV i) (seqp_sos Γ_QMSG))"
proof (rule node_lift_anycast_statelessassm [OF par_rreq_rrep_nsqn_fresh_any_step]) auto
lemma node_anycast_msg_zhops:
shows "(i : opaodv i (⟨i qmsg : R⟩), A) 
(λσ . oarrivemsg (λ_. rreq_rrep_sn) σ, other (λ_ _ True) {i} →)
globala (λ(σ ., a, _). castmsg msg_zhops a)"
by (rule node_lift_anycast StatelessAssm [OF par_anycast_msg_zhops])

lemma node_silent_change_only:
shows "(i : opaodv i (⟨i qmsg : R⟩), A) 
(λσ . oarrivemsg (λ_. rreq_rrep_sn) σ, 
other (λ_ _ True) {i} →)
globala (λ(σ ., a, _). castmsg msg_zhops a)"
proof (rule node_lift_anycast StatelessAssm [OF par_anycast_msg_zhops])

5.11.3 Lift to partial networks
(proof (rule ostep_invariantI, simp (no_asm), rule impI)
fix σ ζ a σ' ζ'
assume or: "((σ, ζ) ∈ ooreachable ((i : opaodv i (⟨i qmsg : R⟩), A) 
(λσ . oarrivemsg (λ_. rreq_rrep_sn) σ, 
other (λ_ _ True) {i} →)
and tr: "((σ, ζ), a, (σ', ζ')) ∈ trans ((i : opaodv i (⟨i qmsg : R⟩), A)"
and "a ≠ τ_n"
from or obtain p R where "ζ = NodeS i p R"
by - (drule node_net_state, metis)
with tr have "((σ, i. NodeS i p R), a, (σ', ζ')) 
in onode_sos (oparp_sos i (trans (opaodv i)) (trans qmsg))"
by simp
thus "σ' i = σ i" using (σ ≠ τ_n)
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 "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_nhops_quality_increases:
shows "opnet (λi. opaodv i (⟨i qmsg : R⟩)) 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_nhops_quality_increases])
fix i R
have "(i : opaodv i (⟨i qmsg : R⟩), 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) ∈ ooreachable ((i : opaodv i (⟨i qmsg : R⟩), A) 
(λσ . oarrivemsg (λ_. rreq_rrep_sn) σ, 
other (λ_ _ True) {i} →)"
and tr: "((σ, s), a, (σ', s')) ∈ trans ((i : opaodv i (⟨i qmsg : R⟩), A)"
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 : opaodv i (⟨i qmsg : R⟩), A) 
(λσ . oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ,
490
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 : opaodv i ⟨⟨ i qmsg : R⟩⟩ ⊨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 : opaodv i ⟨⟨ i qmsg : R⟩⟩ ⊨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

5.11.4 Lift to closed networks

lemma onet_nhop_quality_increases:
  shows "oclosed (opnet (λi. opaodv i ⟨⟨ i qmsg : p⟩⟩) |
  (λσ _ _ _ . 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. opaodv i ⟨⟨ i qmsg : p⟩⟩) |
  (otherwith (op=) (net_tree_ips p) inoclosed, ?U _ →) ?inv"
proof (rule oinvariant_weakenE)
  fix σ σ' :: "ip ⇒ state" and a :: "msg node_action"
  assume "otherwith (op=) (net_tree_ips p) inoclosed σ σ' a"
  thus "oarrivemsg (λσ m. msg_fresh σ m ∧ msg_zhops m) σ σ' a"
  proof (cases a)
    fix ii ni ms
    assume "a = ii _ ni:arrive(ms)"
    moreover with "inoclosed σ a" obtain 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 aodv_openproc: openproc paodv opaodv id
  where "aodv_openproc.initmissing = initmissing"
proof -
  show "openproc paodv opaodv id"
  proof unfold_locales
    fix i :: ip
    have ""{(σ, ζ). (σ i, ζ) ∈ σ_AODV i ∧ (∀j. j ≠ i → σ j ∈ fst ' σ_AODV j)} ⊆ σ_AODV'
    unfolding σ_AODV_def σ_AODV'_def
    proof (rule equalityD1)
show "\(\forall p. \{ (\sigma, \zeta) \mid (\sigma i, \zeta) \in \{(f i, p)\} \wedge (\forall j. j \neq i \rightarrow \sigma j \in \text{fst } \{' \{(f j, p)\}\} = \{(f, p)\}" 
by (rule set_eqI) auto  
qed  
thus "\{(\sigma, \zeta) \mid \sigma \zeta s. s \in \text{init } (\text{paodv } i) 
\wedge (\sigma i, \zeta) = \text{id } s 
\wedge (\forall j. j \neq i \rightarrow \sigma j \in (\text{fst } \text{id}) \ ' \ \text{init } (\text{paodv } j)) \} \subseteq \text{init } (\text{opaodv } i)" 
by simp 
next 
show "\(\forall j. \text{init } (\text{paodv } j) \neq \{\}" 
unfolding \(\sigma_{\text{AODV}}\_\text{def}\) by simp 
next 
fix i s a s' \sigma \sigma' 
assume "\(\sigma i = \text{fst } (\text{id } s)\)" 
and "\(\sigma' i = \text{fst } (\text{id } s')\)" 
and "\((\sigma, a, \sigma') \in \text{trans } (\text{paodv } i)\)" 
then obtain q q' where "\(s = (\sigma i, q)\)" 
and "\(s' = (\sigma' i, q')\)" 
and "\(((\sigma i, q), a, (\sigma' i, q')) \in \text{trans } (\text{paodv } i)\)" 
by (cases s, cases s') auto  
from this(3) have "\(((\sigma, q), a, (\sigma', q')) \in \text{trans } (\text{opaodv } i)\)" 
by simp (rule \text{open_seqp_action} \[\text{OF aodv_wf}\]) 
with \(\langle s = (\sigma i, q) \rangle\) and \(\langle s' = (\sigma' i, q') \rangle\)  
show "\(((\sigma, \text{snd } (\text{id } s)), a, (\sigma', \text{snd } (\text{id } s'))) \in \text{trans } (\text{opaodv } i)\)" 
by simp  
qed  
then interpret op: openproc \text{paodv} \text{opaodv } \text{id} . 
have [simp]: "\(\forall i. (\text{SOME } x. x \in (\text{fst } \text{id}) \ ' \ \text{init } (\text{paodv } i)) = \text{aodv_init } i\)" 
unfolding \(\sigma_{\text{AODV}}\_\text{def}\) by simp 
then unfolding \(\text{openproc.initmissing}\) \text{paodv } \text{id} = \text{initmissing } i" 
unfolding opq.initmissing_def opq.someinit_def initmissing_def 
by (auto split: option.split)  
thus "openproc.initmissing \text{paodv } \text{id} = \text{initmissing}\) .. 
qed 
interpretation \text{aodv.openproc_par_qmsg}: openproc_parq \text{paodv} \text{opaodv } \text{id} \text{qmsg} 
where "\(\text{aodv.openproc_par_qmsg.netglobal } = \text{netglobal}\)" 
and "\(\text{aodv.openproc_par_qmsg.initmissing } = \text{initmissing}\)" 
proof - 
show "openproc_parq \text{paodv } \text{opaodv } \text{id} \text{qmsg}" 
by (unfold_locales) simp 
then interpret opq: openproc_parq \text{paodv} \text{opaodv } \text{id} \text{qmsg} . 
have im: "\(\forall \sigma. \text{openproc.initmissing } (\lambda i. \text{paodv } i \langle \langle \text{qmsg} \rangle \ (\lambda (p, q). (\text{fst } (\text{id } p), \text{snd } (\text{id } p), q)) \sigma = \text{initmissing } \sigma\)" 
unfolding opq.initmissing_def opq.someinit_def initmissing_def 
unfolding \(\sigma_{\text{AODV}}\_\text{def}\) \(\sigma_{\text{QMSG}}\_\text{def}\) by (clarsimp cong: option.case_cong) 
thus "openproc.initmissing \(\lambda i. \text{paodv } i \langle \langle \text{qmsg} \rangle \ (\lambda (p, q). (\text{fst } (\text{id } p), \text{snd } (\text{id } p), q)) \) P \sigma = \text{netglobal } P \sigma\)" 
unfolding opq.netglobal_def \(\sigma_{\text{QMSG}}\_\text{def}\) 
by (clarsimp cong: option.case_cong 
simp del: \text{One_nz_def} 
simp add: \text{fst_initmissing.netmap_default.aodv_init.netlift} 
[symmetric, unfolded initmissing_def])  
thus "openproc.netglobal \(\lambda i. \text{paodv } i \langle \langle \text{qmsg} \rangle \ (\lambda (p, q). (\text{fst } (\text{id } p), \text{snd } (\text{id } p), q)) \) = \text{netglobal}\)" 
by auto 
qed 
lemma \text{net_nhopp_quality_increases}: 
asumes "\text{wf_net_tree } n\)"
shows "closed (pnet (λi. paodv 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. paodv 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 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 σ i
assume sr: "σ ∈ reachable (closed (pnet (λi. paodv i ⟨⟨ qmsg) n)) TT"
  hence "∀ i∈net_tree_ips n. ?inv (fst (initmissing (netgmap fst σ))) i"
  by - (drule invariantD [OF proto],
  simp only: aodv_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. paodv 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 = aodv_init i"
  by simp
thus ?thesis by simp
qed metis
qed

5.11.6 Loop freedom of AODV

theorem aodv_loop_freedom:
  assumes "wf_net_tree n"
shows "closed (pnet (λi. paodv i ⟨⟨ qmsg) n) ⪯ netglobal (λσ. ∀ dip. irrefl ((rt_graph σ dip)⁺))"
using assms by (rule aodv_openproc_par_qmsg.netglobal_weakenE
  [OF net_nhop_quality_increases inv_to_loop_freedom])
end
Bibliography

http://afp.sf.net/entries/AWN.shtml.

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 

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.

wireless mesh networks used for modelling, verifying and analysing AODV. Technical Report 5513, NICTA, 
2013.
