A Process Calculus for Dynamic Networks

Dimitrios Kouzapas, Anna Philippou

A Process Calculus for Dynamic Networks

Dimitrios Kouzapas[1] and Anna Philippou[2]

[1] Imperial College, London, UK. dk208@doc.ic.ac.uk
[2] University of Cyprus, Cyprus. annap@cs.ucy.ac.cy

Abstract. In this paper we propose a process calculus framework for dynamic networks in which the network topology may change as computation proceeds. The proposed calculus allows one to abstract away from neighborhood-discovery computations and it contains features for broadcasting at multiple transmission ranges and for viewing networks at different levels of abstraction. We develop a theory of confluence for the calculus and we use the machinery developed towards the verification of a leader-election algorithm for mobile ad hoc networks.

1 Introduction

Distributed and wireless systems present today one of the most challenging areas of research in computer science. Their high complexity, dynamic nature and features such as broadcasting communication, mobility and fault tolerance, render their construction, description and analysis a challenging task. The development of formal frameworks for describing and associated methodologies for reasoning about distributed systems has been an active area of research for the last few decades. Process calculi, e.g. [7, 8], are one such formalism. Since their inception, they have been extensively studied and they have been extended for modeling and reasoning about a variety of aspects of process behavior including mobility, distribution and broadcasting.

Our goal in this paper is to propose a process calculus in which to be able to reason about mobile ad hoc networks (MANETs) and their protocols. Our proposal, the Calculus for Systems with Dynamic Topology, CSDT, is inspired by works which have previously appeared in the literature [2, 5, 4, 12, 13] on issues such as broadcasting, movement and separating between a node's control and topology information. However, our calculus extends these works by considering two additional MANET features. The first concerns the ability of MANETs to broadcast messages at different transmission levels. This is a standard feature of mobile ad hoc networks; a node may choose to broadcast a message at a high transmission level in order to communicate with a wider set of nodes, as required by a protocol or application, or it may choose to use a low transmission level in order to conserve its power. The second feature concerns that of neighbor discovery. As computation proceeds and nodes move in and out of each other's transmission range, each node attempts to remain aware of its connection topology in order to successfully complete its tasks (e.g. routing). To achieve this, neighbor discovery protocols are implemented and run which typically involve periodically emitting "hello" messages and acknowledging such messages received by one's neighbors. Although it is possible that at various points in time a node does not have the precise information regarding its neighbors, these protocols aim to ensure that the updated

topology is discovered and that the node adjusts its behavior accordingly so that correct behavior is achieved. Thus, when one is called to model and verify a MANET protocol, it is important to take into account that the neighbor-information available to a node may not be correct at all times. To handle this one might model an actual neighbor-discovery protocol in parallel to the algorithm under study and verify the composition of the two. Although such a study would be beneficial towards obtaining a better understanding of the behavior of both protocols, it would turn an already laborious task into a more laborious one and it would impose further requirements on behalf of the modeling language (e.g. to capture timed behaviors).

CSDT allows for reasoning about both of these aspects of behavior. To begin with, it allows nodes to broadcast information at two different transmission levels. The transmission level of a broadcast is encoded as a parameter of broadcasted messages. Regarding the issue of neighborhood discovery, the semantics of CSDT contain rules that mimic the behavior of a neighborhood-discovery protocol: CSDT equips nodes with knowledge of their believed (and not necessarily actual) sets of neighbors which is continuously updated, similarly to the way a neighborhood-discovery algorithm operates and gradually discerns changes in the connectivity of a node. Furthermore, these neighbor sets are accessible from the control part of a node and may affect the flow of the node's execution. A final novelty of CSDT is the introduction of a hiding construct that allows us to observe networks at different levels of abstraction. Since messages in our network descriptions are *broadcasted* over the medium, the notion of channel restriction (or name hiding) becomes irrelevant. Nonetheless, the effect of hiding behaviors remains useful in our setting, especially for analysis purposes. To achieve this, we associate every message with a "type" and we implement hiding by restricting the set of message types which should be observable at the highest level of a process. A similar encapsulation construct has also recently been proposed in [1].

The operational semantics of our calculus is given in terms of a labelled transition system on which we propose a notion of weak bisimulation. Subsequently, we develop a theory of confluence. The notion of confluence was first studied in the context of concurrent systems by Milner in CCS [7] and subsequently in various other settings [14, 10, 11]. Its essence, is that "of any two possible actions, the occurrence of one will never preclude the other". This paper, is the first to consider the theory of confluence in a setting of broadcasting communication. We establish a variety of results including a theorem that allows for compositional reasoning of confluent behavior under the assumption of a stationary topology of a network. We illustrate the utility of these techniques as well as the formalism via the analysis of a leader-election algorithm.

Related Work. Several process calculi have recently been proposed for dynamic networks such as CBS# [9], CMAN [2], CMN [5], CNT [1], CWS [4], TCSW [6], and the ω-calculus [13]. Most of these calculi, support broadcasting communication, message loss and mobility of nodes, with the exception of [4], and explicit location information, with the exception of the ω-calculus, where neighborhood information is captured via groups which can be dynamically created or updated thus modeling dynamic network topologies, and CNT where topology changes are modeled in the semantics as opposed to the syntax of the language. Perhaps closest to CSDT is the CMN process calculus. Differences that exist between the two calculi include (1) the treatment of the notion of

a location (while in CMN locations can be viewed as values in a coordinate system and neighborhood is computed via a metric distance function, in CSDT locations and their interconnections are captured as a graph), (2) the fact that CSDT allows point-to-point communication in addition to broadcasting and (3) CMN caters for lossy communication whereas CSDT does not. Furthermore, as it has already been discussed, CSDT extends all of these frameworks by allowing to broadcast at different transmission levels and by associating nodes with their believed sets of neighbors which are updated by semantic rules that mimic the behavior of a neighbor discovery protocol.

Contribution. The contribution of our work is summarized as follows:

1. We define a new process calculus for reasoning about dynamic networks. This calculus introduces a number of new ideas which have been selected in view of facilitating the modeling and the analysis of mobile ad hoc network protocols.
2. We develop the theory of confluence in the context of our calculus. This is the first such theory for process calculi featuring broadcast communication.
3. We provide a correctness proof of a non-trivial leader-election protocol proposed in [15] for mobile ad hoc networks.

2 The Process Calculus

In our Calculus for Systems with Dynamic Topology, CSDT, we consider a system as a set of nodes operating in space. Each node possesses a physical location and a unique identifier. Movement is modeled as the change in the location of a node, with the restriction that the originating and the destination locations are neighboring locations.

Nodes in CSDT can communicate with each other by broadcasting messages. Broadcasting may take place at different transmission levels as required by an underlying protocol and/or for power-saving purposes. Specifically, in CSDT we consider two transmission levels, a normal level and a high level.

Neighbor discovery, that is, determining which nodes fall within the transmission range of a node, is a building block of network protocols and applications. To facilitate the reasoning about such protocols we embed in the semantics of our calculus rules that mimic the behavior of a neighbor discovery algorithm, which observes changes in the network's topology and, specifically, the departure and arrival of nodes within the normal and high transmission ranges of a node. To achieve this, each node is associated with two sets of nodes, N and H, which are the sets of nodes believed to be within the normal and high transmission ranges of a node, respectively. Thus, we write $P{:}[\![id, \ell, N, H]\!]$, for describing a node running code P with unique identifer id, located at physical location ℓ, believed normal-range and high-range neighbors N, and H.

2.1 The Syntax

We now continue to formalize the above intuitions into the syntax of CSDT. We begin by describing the basic entities of CSDT. We assume a set of node identifiers $\mathcal{I}$ ranged over by id, i, j, and a set of physical locations $\mathcal{L}$ ranged over by ℓ, ℓ' and we say

that two locations ℓ, ℓ' are neighboring locations if $(\ell, \ell') \in Nb$, where $Nb \subseteq \mathcal{L} \times \mathcal{L}$. Furthermore, we assume a set of transmission levels $\{\mathrm{n}, \mathrm{h}\}$ and associated with these levels the functions $\mathrm{range}_{\mathrm{n}} : \mathcal{L} \to 2^{\mathcal{L}}$ and $\mathrm{range}_{\mathrm{h}} : \mathcal{L} \to 2^{\mathcal{L}}$ which, given a location, return the locations that fall within its normal and high transmission levels, respectively. These functions, need not take a symmetric view on locations and may be defined so as to yield unidirectional links. Furthermore, we assume a set of special labels $\mathcal{T}$. Elements of $\mathcal{T}$ are mere keywords appended to messages indicating the message type.

In addition, we assume a set of *terms*, ranged over by e, built over (1) a set of *constants*, ranged over by u, v, (2) a set of *variables* ranged over by x, y, and (3) function applications of the form $f(e_1, \ldots e_n)$ where f is a function from a set of functions (e.g. logical connectives, set operators and arithmetic operators), and the e_i are terms. We say that a term is *closed* if it contains no variables. The evaluation relation $\twoheadrightarrow$ for closed terms is defined in the expected manner. We write $\tilde{r}$ for a tuple of syntactic entities $r_1, \ldots, r_n$. Finally, we assume a set of process constants $\mathcal{C}$, denoted by C, each with an associated definition of the form $C\langle\tilde{x}\rangle \stackrel{\mathrm{def}}{=} P$, where P may contain occurrences of C, as well as other constants. Based on these basic entities, the syntax of of CSDT is given in Table 1, where $T \subseteq \mathcal{T}$.

Table 1. The Syntax

Actions:	η	$::= \overline{b}(w, t, \tilde{v}, tl)$	broadcast
		$\mid \; r(t, \tilde{x})$	input
Processes:	P	$::= \mathbf{0}$	Inactive Process
		$\mid \; \eta.P$	Action Prefix
		$\mid \; P_1 + P_2$	Nondeterministic Choice
		$\mid \; \mathsf{cond}\ (e_1 \rhd P_1, \ldots, e_n \rhd P_n)$	Conditional
		$\mid \; C\langle\tilde{v}\rangle$	Process Constant
Networks:	M	$::= \mathbf{0}$	
		$\mid \; P{:}\sigma$	Located Node
		$\mid \; M_1 \mid M_2$	Parallel Composition
		$\mid \; M \backslash T$	Restriction
Interfaces:	σ	$::= [\![id, \ell, N, H]\!]$	

There are two types of actions in CSDT. A broadcast action $\overline{b}(w, t, \tilde{v}, tl)$ is a transmission at transition level $tl \in \{\mathrm{n}, \mathrm{h}\}$, of type $t \in \mathcal{T}$ of the tuple $\tilde{v}$ with intended recipients w, where $'-'$ denotes that the message is intended for all neighbors of the transmitting node and $j \in \mathcal{I}$ denotes that it is intended solely for node j. An input action $r(t, \tilde{x})$ represents a receipt of a message $\tilde{x}$ of type t. In turn, a process can then be inactive, an action-prefixed process, the nondeterministic choice between two processes, a process constant or a conditional process. The conditional process $\mathsf{cond}\ (e_1 \rhd P_1, \ldots, e_n \rhd P_n)$ presents the conditional choice between a set of processes: it behaves as P_i, where i is the smallest integer for which e_i evaluates to true.

On the other hand, networks are built on the basis of located processes, $P{:}\sigma$, where σ is the node's topology information which we call its *interface*. An interface σ contains the node identifier id, its location ℓ as well as its normal-range and high-range neighbors N and H, according to its current knowledge. We allow the control part of a located process $P{:}\sigma$, namely P, to access information mentioned in the interface σ, by using the special labels id, l, N and H, thus allowing the control part of a process to express dependencies on the node's neighborhood information. For example, an expression "$4 \in \mathrm{N}$" occurring within $P : [\![1, \ell, \{2\}, \{2, 3\}]\!]$ is evaluated as "$4 \in \{2\}$".

Thus, a network can be an inactive network $\mathbf{0}$, a located node $P{:}\sigma$, a parallel composition of networks $M_1 \mid M_2$, or a restricted network $M\backslash T$. In $M\backslash T$ the scope of messages of all types in T is restricted to network M: components of M may exchange messages of these types to interact with one another but not with M's environment. To avoid name collisions on types, we consider α-conversion on processes as the renaming of types that are bound by some restriction and we say that P and Q are α-equivalent, $P \equiv_\alpha Q$, if P and Q differ by a renaming of bound types.

In what follows, given an interface σ, we write $\mathrm{l}(\sigma)$ and $\mathrm{id}(\sigma)$ for the location and the identifier mentioned in σ, respectively. Moreover, we use $\mathrm{types}(M)$ to denote the set of all types occurring in activities of M.

2.2 The Semantics

The semantics of CSDT is defined in terms of a structural congruence relation $\equiv$, found in Table 2, and a structural operational semantics, given in Tables 3 and 4.

Table 2. Structural congruence relation

(N1) $M \equiv M \mid 0$	(N5) $M\backslash T - \{t\} \equiv M\backslash T$ if $t \notin \mathrm{types}(M)$
(N2) $M_1 \mid M_2 \equiv M_2 \mid M_1$	(N6) $M\backslash T \equiv (M\backslash T - \{t\})\backslash\{t\}$
(N3) $(M_1 \mid M_2) \mid M_3 \equiv M_1 \mid (M_2 \mid M_3)$	(N7) $M_1 \equiv M_2$ if $M_1 \equiv_\alpha M_2$
(N4) $M_1\backslash\{t\} \mid M_2 \equiv (M_1 \mid M_2)\backslash\{t\}$ if $t \notin \mathrm{types}(M_2)$	

The rules of Table 3 describe the behavior of located processes in isolation whereas the rules in Table 4 the behavior of networks. A transition of P (or M) has the form $P \xrightarrow{\alpha} P'$, specifying that P can perform action α and evolve into P' where α can have one of the following forms:

- $\overline{b}(w, t, \tilde{v}, tl, \ell)$ denotes a broadcast to recipients w of a message $\tilde{v}$ of type t at transmission level tl, taking place at location ℓ.
- $r(id, t, \tilde{v}, \ell)$ denotes a receipt by node id of a message $\tilde{v}$ of type t, taking place at location ℓ.

– $r?(id, t, \tilde{v}, \ell)$ denotes an advertisement by node id that it is willing to receive a message $\tilde{v}$ of type t, at location ℓ.
– τ and μ denote two types of unobservable actions in the calculus. Action τ is associated with the effect of restriction (rule (Hide2), Table 4) and μ is associated with the movement of nodes (rule (L6), Table 3) as well as with updates of neighborhood information (rules (InSN), (InSH), (OutSN) and (OutSH), Table 4).

We let *Act* denote the set of all actions and let α and β range over *Act* and we write $\mathrm{type}(\alpha)$ for the type of an $\alpha \in Act - \{\tau, \mu\}$.

Table 3. Transition rules for located nodes

(L1) $(\overline{b}(w,t,\tilde{v},tl).P){:}\sigma \overset{\overline{b}(w,t,\tilde{v},tl,\mathrm{l}(\sigma))}{\longrightarrow} P{:}\sigma$ (L2) $(r(t,\tilde{x}).P){:}\sigma \overset{r(\mathrm{id}(\sigma),t,\tilde{v},\mathrm{l}(\sigma))}{\longrightarrow} P\{\tilde{v}/\tilde{x}\}{:}\sigma$

(L3) $\dfrac{[P\{\tilde{v}/\tilde{x}\}]{:}\sigma \overset{\alpha}{\longrightarrow} P'{:}\sigma}{[C\langle\tilde{v}\rangle]{:}\sigma \overset{\alpha}{\longrightarrow} P'{:}\sigma} \quad C\langle\tilde{x}\rangle \overset{\mathrm{def}}{=} P$ (L4) $\dfrac{P_i{:}\sigma \overset{\alpha}{\longrightarrow} P_i'{:}\sigma}{(P_1+P_2){:}\sigma \overset{\alpha}{\longrightarrow} P_i'{:}\sigma}, \ i \in \{1,2\}$

(L5) $\dfrac{P_i{:}\sigma \overset{\alpha}{\longrightarrow} P_i'{:}\sigma}{(\mathsf{cond}\ (e_1 \rhd P_1, \ldots, e_n \rhd P_n)){:}\sigma \overset{\alpha}{\longrightarrow} P_i'{:}\sigma} \quad e_i \twoheadrightarrow \mathsf{true}, \forall j < i, e_j \twoheadrightarrow \mathsf{false}$

(L6) $\dfrac{(\ell,\ell') \in Nb}{P{:}[\![i,\ell,N,H]\!] \overset{\mu}{\longrightarrow} P{:}[\![i,\ell',N,H]\!]}$

Moving our attention to the rules of Table 4, we point out that the first four rules implement the underlying neighborhood discovery protocol. Nodes which are within the normal and high transmission ranges of a located process are included in the appropriate sets in its interface and, similarly, nodes which have exited these transmission ranges are removed. In all four cases a μ internal action takes place.

The two (BrC) rules which follow give semantics to broadcasting within the language. They employ the compatibility function comp, where $\mathrm{comp}(w,i)$ is evaluated to true only if identifier i is compatible with intended recipient w: $\mathrm{comp}(w,i) = (w =' -' \vee\ w = i)$. Axiom (BrC1) specifies that, if a broadcast is available and there exists a compatible recipient within the range of the broadcast, the message is received and the broadcast message is propagated. If there is no appropriate receiver then, again, the broadcast is propagated (BrC2). Note that rule (BrC2) (as well as (Rec)) is defined in terms of the inability of executing an action via the use of relation $\overset{\alpha}{\nrightarrow}$, where $P \overset{\alpha}{\nrightarrow}$, if $\neg(P \overset{\alpha}{\longrightarrow} P')$ for any P'.

Moving on to rule (Rec), we observe that a network can advertise the fact that it may receive an input. This is necessary, otherwise an inactive network and a network such as $M \overset{\mathrm{def}}{=} [r(t,\tilde{x}).P]{:}\sigma_1 \mid [r(t,\tilde{y}).Q]{:}\sigma_2$ would have exactly the same transition systems when clearly they would yield distinct behaviors when placed in parallel with a network such as $M' = [\overline{b}(-,t,\tilde{v},\mathrm{n}).S]{:}\sigma$ (affecting compositionality in the calculus). Now, if rule (Rec) was enunciated via a normal receive action instead of $r?$, we would have: $M \overset{r(\mathrm{id}(\sigma_1),t,\tilde{v},\mathrm{l}(\sigma_1))}{\longrightarrow} [P\{\tilde{v}/\tilde{x}\}]{:}\sigma_1 \mid [r(t,\tilde{y}).Q]{:}\sigma_2$ and subsequently

Table 4. Transition rules for networks

(InSN) $$\frac{\mathrm{l}(\sigma) \in \mathrm{range_n}(\ell),\ \mathrm{id}(\sigma) \notin N}{P{:}[\![id, \ell, N, H]\!] \mid Q{:}\sigma \xrightarrow{\mu} P{:}[\![id, \ell, N \cup \{\mathrm{id}(\sigma)\}, H]\!] \mid Q{:}\sigma}$$

(OutSN) $$\frac{\mathrm{l}(\sigma) \notin \mathrm{range_n}(\ell),\ \mathrm{id}(\sigma) \in N}{P{:}[\![id, \ell, N, H]\!] \mid Q{:}\sigma \xrightarrow{\mu} P{:}[\![id, \ell, N - \{\mathrm{id}(\sigma)\}, H]\!] \mid Q{:}\sigma}$$

(InSH) $$\frac{\mathrm{l}(\sigma) \in \mathrm{range_h}(\ell),\ \mathrm{id}(\sigma) \notin H}{P{:}[\![id, \ell, N, H]\!] \mid Q{:}\sigma \xrightarrow{\mu} P{:}[\![id, \ell, N, H \cup \{\mathrm{id}(\sigma)\}]\!] \mid Q{:}\sigma}$$

(OutSH) $$\frac{\mathrm{l}(\sigma) \notin \mathrm{range_h}(\ell),\ \mathrm{id}(\sigma) \in H}{P{:}[\![id, \ell, N, H]\!] \mid Q{:}\sigma \xrightarrow{\mu} P{:}[\![id, \ell, N, H - \{\mathrm{id}(\sigma)\}]\!] \mid Q{:}\sigma}$$

(BrC1) $$\frac{M_1 \xrightarrow{\overline{b}(w,t,\tilde{v},tl,\ell)} M_1', M_2 \xrightarrow{r(id,t,\tilde{v},\ell')} M_2', \mathrm{comp}(w, id), \ell' \in \mathrm{range}_{tl}(\ell)}{M_1 \mid M_2 \xrightarrow{\overline{b}(w,t,\tilde{v},tl,\ell)} M_1' \mid M_2'}$$

(BrC2) $$\frac{M_1 \xrightarrow{\overline{b}(w,t,\tilde{v},tl,\ell)} M_1' \text{ and } M_2 \xrightarrow{r(id,t,\tilde{v},\ell')\not\rightarrow} \forall id, \ell' \cdot \mathrm{comp}(w, id), \ell' \in \mathrm{range}_{tl}(\ell)}{M_1 \mid M_2 \xrightarrow{\overline{b}(w,\ell,l,t,\tilde{v})} M_1' \mid M_2}$$

(Rec) $$\frac{M_1 \xrightarrow{r(id,\ell,t,\tilde{v})} M_1' \text{ and } M_2 \xrightarrow{\overline{b}(w,t,\tilde{v},tl,\ell')\not\rightarrow} \forall w, \ell', tl \cdot \mathrm{comp}(w, id), \ell \in \mathrm{range}_{tl}(\ell')}{M_1 \mid M_2 \xrightarrow{r?(id,\ell,t,\tilde{v})} M_1' \mid M_2}$$

(Hide1) $$\frac{M \xrightarrow{\alpha} M' \text{ and } \mathrm{type}(\alpha) \notin T}{M \backslash T \xrightarrow{\alpha} M' \backslash T}$$

(Hide2) $$\frac{M \xrightarrow{\alpha} M' \text{ and } \mathrm{type}(\alpha) \in T}{M \backslash T \xrightarrow{\tau} M' \backslash T}$$

(Par) $$\frac{M_1 \xrightarrow{\alpha} M_1', \quad \alpha \in \{\tau, \mu\}}{M_1 \mid M_2 \xrightarrow{\alpha} M_1' \mid M_2}$$

(Struct) $$\frac{M_1 \equiv M_2, M_2 \xrightarrow{\alpha} M_2', M_2' \equiv M_1'}{M_1 \xrightarrow{\alpha} M_1'}$$

$M \mid M' \xrightarrow{\overline{b}(-,t,\tilde{v},\mathrm{n},\mathrm{l}(\sigma))} ([P\{\tilde{v}/\tilde{x}\}]{:}\sigma_1 \mid [r(t, \tilde{y}).Q]{:}\sigma_2) \mid S{:}\sigma$, In this way only one of the two components of M ends up receiving the broadcasted message of M' which is not what one would expect of a broadcasting communication. To achieve the correct interpretation, only located nodes can emit an $r(\ldots)$ action (see (Rec), Table 3) and to obtain the transition of $M \mid M'$ we would employ structural congruence and the facts that (1) $M \mid M' \equiv M_1 = [r(t, \tilde{x}).P]{:}\sigma_1 \mid ([r(t, \tilde{y}).Q]{:}\sigma_2 \mid [\overline{b}(-, t, \tilde{v}, \mathrm{n}).S]{:}\sigma)$, (2) $M_1 \xrightarrow{\overline{b}(-,t,\tilde{v},\mathrm{n},\mathrm{l}(\sigma))} [P\{\tilde{v}/\tilde{x}\}]{:}\sigma_1 \mid ([Q\{\tilde{v}/\tilde{y}\}]{:}\sigma_2 \mid S{:}\sigma)$, to obtain, by rule (Struct), that $M \mid M' \xrightarrow{\overline{b}(-,t,\tilde{v},\mathrm{n},\mathrm{l}(\sigma))} ([P\{\tilde{v}/\tilde{x}\}]{:}\sigma_1 \mid [Q\{\tilde{v}/\tilde{y}\}]{:}\sigma_2) \mid S{:}\sigma$.

Finally, rules (Hide1) and (Hide2) specify that the effect of restricting a set of types T within a process is to transform all actions of these types into internal actions.

2.3 Bisimulation and Confluence

In the next section we build some machinery for reasoning about broadcasting networks. Due to lack of space, all proofs from this section are omitted. The complete exposition can be found in [3]. First, let us recall that M' is a *derivative* of M, if there exist actions $\alpha_1, \ldots, \alpha_n, n \geq 0$, such that $M \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} M'$. Moreover, we define weak transitions as follows: given an action α we write $M \Longrightarrow M'$ for $M(\xrightarrow{\tau,\mu})^* M'$, $M \stackrel{\alpha}{\Longrightarrow} M'$ for $M \Longrightarrow \xrightarrow{\alpha} \Longrightarrow M'$, and $M \stackrel{\hat{\alpha}}{\Longrightarrow} M'$ for $M \Longrightarrow M'$ if $\alpha \in \{\tau, \mu\}$, $M \stackrel{r(id,\ell,t,\tilde{v})}{\Longrightarrow} M'$ if $\alpha = r?(id, \ell, t, \tilde{v})$, and $M \stackrel{\alpha}{\Longrightarrow} M'$ otherwise.

The first useful tool which accompanies CSDT is a notion of observational equivalence:

Definition 1. *Bisimilarity* is the largest symmetric relation, denoted by $\approx$, such that, if $M_1 \approx M_2$ and $M_1 \xrightarrow{\alpha} M_1'$, there exists M_2' such that $M_2 \stackrel{\hat{\alpha}}{\Longrightarrow} M_2'$ and $M_1' \approx M_2'$.

We may prove that bisimilarity is a congruence relation. It is also worth pointing out that we may establish the following property of mobile nodes pertaining to their ubiquitous nature, namely:

Lemma 1. For any process P, $id \in \mathcal{I}$, $\ell, \ell' \in \mathcal{L}$ and $N, H \subseteq \mathcal{L}$, if $(\ell, \ell') \in Nb^+$, where Nb^+ is the transitive closure of relation *Nb*, then $P[\![id, \ell, N, H]\!] \approx P[\![id, \ell', N, H]\!]$.

We now turn to consider the notions of determinacy and confluence in our calculus. These make use of the following notation: given actions α and α' we write $\alpha \bowtie \alpha'$ if α and α' differ only in their specified location, i.e. $\alpha' = \alpha[\ell'/\ell]$, for some ℓ and ℓ'.

Definition 2. M is *determinate* if, for every derivative M' of M and for all actions α, α', $\alpha \bowtie \alpha'$, whenever $M' \xrightarrow{\alpha} M_1$ and $M' \stackrel{\hat{\alpha'}}{\Longrightarrow} M_2$ then $M_1 \approx M_2$.

This definition makes precise the requirement that, when an experiment is conducted on a network, it should always lead to the same state up to bisimulation. This is irrespective of the location at which the action is taking place which explains the use of the $\bowtie$ operator. As an example, consider networks

$$M_1 \stackrel{\text{def}}{=} \overline{b}(-, t, \tilde{v}, \mathrm{n}).\mathbf{0}{:}[\![1, \ell, \{2\}, \{2\}]\!] \qquad M_1' \stackrel{\text{def}}{=} \overline{b}(-, t, \tilde{v}, \mathrm{n}).\mathbf{0}{:}[\![1, \ell', \{2\}, \{2\}]\!]$$
$$M_2 \stackrel{\text{def}}{=} (r(t, \tilde{x}).\overline{b}(-, s, \tilde{x}, \mathrm{h}).\mathbf{0},){:}[\![2, \ell, \{1\}, \{1\}]\!]$$

We observe that M_1 and M_2 are determinate, whereas $M_1 \mid M_2$ is not: Assuming that $\ell \notin \mathrm{range}_n(\ell')$, $M_1 \mid M_2 \xrightarrow{\overline{b}(-,t,\tilde{v},\mathrm{n},\ell)} M \equiv \overline{b}(-, s, \tilde{x}, \mathrm{h}).\mathbf{0}{:}[\![2, \ell, \ldots]\!]$ and $M_1 \mid M_2 \xrightarrow{\mu} M_1' \mid M_2 \xrightarrow{\overline{b}(-,t,\tilde{v},\mathrm{n},\ell')} M' \equiv M_2$, where $M \not\approx M'$.

A conclusion that we may draw from this example is that determinacy is not preserved by parallel composition. We may in fact show that determinacy is preserved by prefix and conditional statements. In order to strengthen determinacy into a notion preserved by a wider range of operators, and in particular parallel composition, Milner [7] introduced the notion of confluence. According to the definition of [7], a CCS process P is *confluent* if it is determinate and, for each of its derivatives Q and distinct actions α,

β, given the transitions to $Q \xrightarrow{\alpha} Q_1$ and $Q \xRightarrow{\beta} Q_2$, the diamond property is satisfied, that is, there exist transitions $Q_1 \xRightarrow{\beta} Q_1'$ and $Q_2 \xRightarrow{\alpha} Q_2'$ such that $Q_1' \approx Q_2'$.

In the context of our work, we observe that the nature of broadcasting communication allows to enunciate confluence through a weaker treatment of input actions. In particular, given the fact that the emission of messages by network nodes is continuously enabled and is not blocked by the availability of a receiver, a network that can receive two distinct inputs need not satisfy the diamond property with respects to these inputs; what matters is that the network can reach equivalent states after each of the two inputs, either by executing the other input or not.

Definition 3. M is *confluent* if it is determinate and, for each of its derivatives M' and distinct actions α, β, where $\neg(\alpha \bowtie \beta)$, whenever $M' \xrightarrow{\alpha} M_1$ and $M' \xRightarrow{\beta} M_2$ then,

1. if $\alpha = r(id, t, \tilde{u}, \ell)$ and $\beta = r(id, t, \tilde{v}, \ell')$, then, either (1) $M_1 \approx M_2$, or (2) $M_1 \xRightarrow{\beta} M_1' \approx M_2$, or (3) $M_2 \xRightarrow{\alpha} M_2' \approx M_1$, or (4) $M_1 \xRightarrow{\beta} M_1'$, $M_2 \xRightarrow{\alpha} M_2'$, and $M_1' \approx M_2'$,
2. otherwise, there are M_1' and M_2' such that $M_2 \xRightarrow{\hat{\alpha}} M_2'$, $M_1 \xRightarrow{\hat{\beta}} M_1'$ and $M_1' \approx M_2'$.

We note that the first clause of the definition captures the four possibilities for bringing together two input actions by receiving further input after the first input, or not.

We may see that bisimilarity preserves confluence. Furthermore, confluent networks possess an interesting property regarding internal actions. We define a network M to be τ-*inert* if, for each derivative M_1 of M, if $M_1 \xrightarrow{\tau} M_2$ or $M_1 \xrightarrow{\mu} M_2$, then $M_1 \approx M_2$. By a generalization of the proof in CCS, we obtain:

Lemma 2. If M is confluent then M is τ-inert.

Although confluence is preserved by more operators than determinacy, namely the restriction operator, it is still not preserved by parallel composition. The counter-example provided in the case of determinacy is still valid: while M_1 and M_2 are confluent, $M_1 \mid M_2$ is not.

The main obstacle in establishing the compositionality of confluence with respect to parallel composition, as well as other operators, is that of node mobility. In what follows we extend our study of confluence in the context of stationary CSDT systems, that is, systems in which there is no movement of nodes (i.e. no μ action). The benefits of this study are twofold. On one hand, we develop a theory of confluence for broadcasting systems which turns out to be both simple as well as compositional. On the other hand, this theory remains useful in the context of mobility since, during the verification of ad hoc network systems, correctness criteria focus on the behavior of systems once their topology remains stable for a sufficient amount of time.

We begin by defining a new notion of weak transition that permits τ actions and excludes μ actions. Specifically, we write $M \Longrightarrow_s M'$, $M \xRightarrow{\alpha}_s M'$ and $M \xRightarrow{\hat{\alpha}}_s M'$ for the subsets of relations $M \Longrightarrow M'$, $M \xRightarrow{\alpha} M'$ and $M \xRightarrow{\hat{\alpha}} M'$ where no $\xrightarrow{\mu}$ are present. In a similar vein, the new notion of bisimilarity, S-bisimilarity matches the behavior of equivalent systems excluding μ-actions:

Definition 4. *S-Bisimilarity* is the largest symmetric relation, denoted by $\approx_s$, such that, if $M_1 \approx_s M_2$ and $M_1 \stackrel{\alpha}{\longrightarrow} M_1'$, $\alpha \in Act - \{\mu\}$, there exists M_2' such that $M_2 \stackrel{\hat{\alpha}}{\Longrightarrow}_s M_2'$ and $M_1' \approx_s M_2'$.

It is easy to see that $\approx \subset \approx_s$ and that S-bisimilarity is a congruence relation. Based on the notion of S-bisimulation, we may define the notions of S-determinacy and S-confluence via variations of Definitions 2 and 3 which replace $\stackrel{a}{\Longrightarrow}$, $\approx$, and *Act* by $\stackrel{\alpha}{\Longrightarrow}_s$, $\approx_s$, and $Act - \{\mu\}$. We may see that S-bisimilarity preserves S-confluence and that S-confluent networks possess the property that their observable behavior remains unaffected by τ actions. In particular, we define a network M to be τ_s-*inert* if, for each derivative M_1 of M, if $M_1 \stackrel{\tau}{\longrightarrow} M_2$, then $M_1 \approx_s M_2$ and we may prove:

Lemma 3. If M is S-confluent then M is τ_s-inert.

We prove that S-confluence is preserved by most CSDT operators including that of parallel composition.

Lemma 4.

1. If $P{:}\sigma$ is S-confluent then so are $(\tau.P){:}\sigma$ and $(\bar{b}(w,t,\tilde{x},tl).P){:}\sigma$.
2. If $P_i{:}\sigma$, $1 \leq i \leq n$, are S-confluent, then so is $\mathsf{cond}\ (e_1 \rhd P_1, \ldots, e_n \rhd P_n){:}\sigma$.
3. If M is S-confluent, then so is $M\backslash T$.
4. If M_1 and M_2 are S-confluent then so is $M_1 \mid M_2$.

3 Specification and Verification of a Leader-Election Algorithm

3.1 The algorithm

The algorithm we consider for our case study is the distributed leader-election algorithm presented in [15]. It operates on an arbitrary topology of nodes with distinct identifiers and it elects as the leader of the network the node with the maximum identifier.

We first describe the static version of the algorithm which assumes that no topology changes are possible. We then proceed to extend this description to the mobile setting. In brief, the static algorithm operates as follows. In its initial state, a network node may initiate a leader-election computation (note that more than one node may do this) or accept leader-election requests from its neighbors. Once a node initiates a computation, it triggers communication between the network nodes which results into the creation of a spanning tree of the graph: each node picks as its father the node from which it received the first request and forwards the request to its remaining neighbors. Each node awaits to receive from each of its children the maximum identifier of the subtree at which they are rooted and, then, forward it to its father. Naturally, the root will receive the maximum identifier of the entire computation tree which is the elected leader. The leader is then flooded to the entire network.

Note that if more than one node initiates a leader-election computation then only one computation survives. This is established by associating each computation with a source identifier. Whenever a node already in a computation receives a request for a computation with a greater source, it abandons its original computation and it restarts a computation with this new identifier.

In the mobile setting, it is easy to observe that with node mobility, crashes and failures as well as network partitions and merges, the above algorithm is inadequate. To begin with, let us note that once a leader is elected it emits so-called *heartbeat* messages to the environment, which are essentially messages sent at a high transmission level. The absence of a heartbeat message from its leader triggers a node to initiate a new computation of a leader. Note that such an absence may just indicate that the node is outside of the high transmission range of the leader even though they belong to the same connected component of the network. However, this does not affect the correctness of the algorithm. Based on this extension, computation proceeds as described by the static algorithm with the exception of the following fine points:

- **Losing a child node.** If a node loses a child then it removes the child from the set of nodes from which it expects an acknowledgement and continues its computation.
- **Losing a parent node.** If a node loses its father then it assigns itself the role of the root of its subtree and continues its computation.
- **Partition Merges.** If two components of the system meet each other, they each proceed with their computation (if they are still computing a leader) ignoring any invitations to join the other's computation. Once they elect their individual leaders and start flooding their results, each adopts the leader with the largest identifier. An exception to this is the case when the two nodes that have come into contact have the same previous-leader field (a piece of information maintained in the mobile version of the algorithm), in which case they proceed as with the static case with the highest valued-computation being the winner.

3.2 Specification of the Protocol

In this subsection we model the algorithm in CSDT. We assume that messages exchanged by the network nodes must be accompanied by one of the following types:

- $\mathbf{election}$: used when a node invites another to join its computation.
- $\mathbf{ack1}$: used to notify the recipient that the sender has agreed to enter its computation and commits to forward the maximum identifier among its downward nodes.
- $\mathbf{ack0}$: used to notify the recipient that the sender has not agreed to be one of its children.
- $\mathbf{leader}$: used to announce the elected leader of a computation during the flooding process.
- $\mathbf{reply}$: used to announce the computation in which a node participates. Such messages are important for the following reason: If a node x departs from its location, enters a new computation and returns to its previous location before its initial neighbors notice its departure, these reply messages will notify initial neighbors that x is no longer part of their computation and thus they will no longer expect x to reply.
- $\mathbf{hbeat}$: a message of this type is emitted by a leader node.

In its initial state the algorithm is modeled by the process S consisting of a set of nodes who possess a leader (although this leader may be outside of their range).

$$\mathrm{S} \stackrel{\mathrm{def}}{=} (\prod_{k \in K} \mathrm{Elected}\langle b_k, s_k, lead_k \rangle{:}\sigma_k) \backslash T,$$

where $T = \{\mathbf{election}, \mathbf{ack0}, \mathbf{ack1}, \mathbf{leader}, \mathbf{reply}\}$. Thus, we restrict all but $\mathbf{hbeat}$ messages emitted by leader nodes which are the messages we wish to observe. Based on these messages we will subsequently express the correctness criterion of the algorithm.

The description of the behavior of nodes can be found in Figure 1. To begin with, a node in $\mathrm{Elected}$ possesses a leader $lead$, a source s which records the computation in which it has participated to elect its leader, and a status b which records whether or not it needs to broadcast its leader. Note that a leader node (i.e. a node with $id = lead$) regularly sends a heartbeat to notify its heartbeat neighbors of its availability. We may see in Figure 1 that if a process $\mathrm{Elected}$ receives an election message from one of its neighbors or, if it observes the absence of its leader (see condition $lead \notin \mathrm{H}$), it enters InComp mode, wherein it begins its quest for a new leader.

The InComp process has a number of parameters: c records whether the node should broadcast a leader election invitation to its neighbors and f is the node's father to which eventually an $\mathbf{ack1}$ message needs to be returned (unless the node itself is the root of the tree). src and $lead$ are the computation's source and previous leader, whereas max is the maximum identifier observed by the node. Sets R and A record the neighbors of the node that are expected to reply and the neighbors to which an $\mathbf{ack0}$ should be returned, respectively. Note that these sets are regularly updated according to the node's interface: we write $R' = R \cap \mathrm{N}$, $A' = A \cap \mathrm{N}$ and $f' = f$, if $f \in \mathrm{N}$, and *NULL*, otherwise. It is worth noting that at these points (as well as others, e.g. "$f = \mathrm{id}$", "$lead \in \mathrm{H}$"), the ability of referring to the node's interface plays a crucial role for the specification of the protocol. Furthermore, the fact that sets N and H are the *believed* sets of neighbors and may contain errors is realistic and it allows us to explore the correctness of the protocol in its full generality.

Finally, node $\mathrm{Leader}\langle f, src, max, lead\rangle$ awaits to be notified of the component's leader by its father f. It recalls its computation characterized by its source and previous leader (src and $lead$) as well as the maximum node it has observed from its downstream nodes, max. In case of the loss of its father, the node elects this maximum node as the leader of the component.

3.3 Verification of the Protocol

The aim of our analysis is to establish that after a finite number of topology changes, every connected component of the network, where connectedness is defined in terms of the normal transmission range, will elect the node with the maximum identifier as its leader. We consider an arbitrary derivative of S, namely S_1, where we assume that for all nodes, sets N and H are consistent with the network's state, and we show:

Theorem 1. $S_1 \approx_s (\prod_{k \in K} \mathrm{Elected}\langle 1, s, max_k\rangle{:}\sigma_k)\backslash T$ *where* max_k *is the maximum node identifier in the connected component of node* k*.*

In words, our correctness criterion states that, eventually, assuming nodes stop moving, all nodes will learn a leader which is the node with the maximum identifier within their connected component. We proceed to sketch the proof of the result which can be found in its entirety in [3].

Sketch of Proof of Theorem 3
Let us consider an arbitrary derivative of S. This may be viewed as the composition of

$$\mathrm{Elected}\langle 0, src, lead\rangle \stackrel{\text{def}}{=} \overline{b}(-, \mathbf{leader}, \langle lead\rangle, \mathrm{n}).\, \mathrm{Elected}\langle 1, src, lead\rangle$$

$$\begin{aligned}
&\mathrm{Elected}\langle 1, src, lead\rangle \stackrel{\text{def}}{=}\\
&\quad \mathsf{cond}\ (lead = \mathrm{id} \rhd \overline{b}(-, \mathbf{hbeat}, \langle lead\rangle, \mathrm{h}).\mathrm{Elected}\langle 1, src, lead\rangle,\\
&\qquad lead \notin \mathrm{H} \rhd \mathrm{InComp}\langle 0, \mathrm{id}, \mathrm{id}, \mathrm{N}, \emptyset, \mathrm{id}, lead\rangle)\\
&\quad + r(\mathbf{leader}, \langle lead'\rangle).\ \mathsf{cond}\ (lead < lead' \rhd \mathrm{Elected}\langle 0, src, lead'\rangle,\\
&\qquad\qquad true \rhd \mathrm{Elected}\langle 1, src, lead\rangle)\\
&\quad + \overline{b}(-, \mathbf{reply}, \langle \mathrm{id}, s\rangle, \mathrm{n}).\, \mathrm{Elected}\langle 1, src, lead\rangle\\
&\quad + r(\mathbf{election}, \langle j, l, s\rangle).\ \mathsf{cond}\ (l = lead \rhd \mathrm{InComp}\langle 0, j, s, \mathrm{N} - \{j\}, \emptyset, \mathrm{id}, lead\rangle,\\
&\qquad\qquad true \rhd \mathrm{Elected}\langle 1, src, lead\rangle)
\end{aligned}$$

$$\begin{aligned}
&\mathrm{InComp}\langle c, NULL, src, R, A, max, lead\rangle \stackrel{\text{def}}{=} \mathrm{InComp}\langle c, \mathrm{id}, src, R', A', max, lead\rangle\\
&\mathrm{InComp}\langle 1, f, src, \emptyset, \emptyset, max, lead\rangle \stackrel{\text{def}}{=}\\
&\quad \mathsf{cond}\ (f = \mathrm{id} \rhd \mathrm{Elected}\langle 0, src, max\rangle,\\
&\qquad true \rhd \overline{b}(f, \mathbf{ack1}, \langle \mathrm{id}, scr, max\rangle, \mathrm{n}).\, \mathrm{Leader}\langle f, src, max, lead\rangle)\\
&\mathrm{InComp}\langle c, f, src, R, A, max, lead\rangle \stackrel{\text{def}}{=}\\
&\quad \mathsf{cond}\ (c = 0 \rhd \overline{b}(-, \mathbf{election}, \langle \mathrm{id}, lead, scr\rangle, \mathrm{n}).\, \mathrm{InComp}\langle 1, f', scr, R', A', max, lead\rangle)\\
&\quad + \textstyle\sum_{j\in A} \overline{b}(j, \mathbf{ack0}, \langle \mathrm{id}\rangle, \mathrm{n}).\, \mathrm{InComp}\langle 1, f', scr, R', A' - \{j\}, max, lead\rangle\\
&\quad + r(\mathbf{election}, \langle j, l, s\rangle).\\
&\qquad \mathsf{cond}\ (l = lead \wedge s > src \rhd \mathrm{InComp}\langle 0, j, s, \mathrm{N} - \{j\}, \emptyset, max, lead\rangle,\\
&\qquad\quad l = lead \wedge s = scr \rhd \mathrm{InComp}\langle c, f', scr, R', A' \cup \{j\}, max, lead\rangle,\\
&\qquad\quad true \rhd \mathrm{InComp}\langle c, f, scr, R', A', max, lead\rangle)\\
&\quad + r(\mathbf{ack0}, \langle j\rangle).\, \mathrm{InComp}\langle 1, f', src, R' - \{j\}, A', max, lead\rangle\\
&\quad + r(\mathbf{ack1}, \langle j, s, max'\rangle).\\
&\qquad \mathsf{cond}\ (s = src \wedge max' > max \rhd \mathrm{InComp}\langle c, f', src, R' - \{j\}, A', max', lead\rangle,\\
&\qquad\quad s = src \wedge max' \leq max \rhd \mathrm{InComp}\langle c, f', src, R' - \{j\}, A', max, lead\rangle,\\
&\qquad\quad true \rhd \mathrm{InComp}\langle c, f', src, R' - \{j\}, A', max, lead\rangle)\\
&\quad + r(\mathbf{leader}, \langle l\rangle).\, \mathrm{InComp}\langle c, f', src, R', A', max, lead\rangle\\
&\quad + r(\mathbf{reply}, \langle j, s\rangle).\\
&\qquad \mathsf{cond}\ (src \neq s \rhd \mathrm{InComp}\langle c, f', src, R' - \{j\}, A' - \{j\}, max, lead\rangle,\\
&\qquad\quad true \rhd \mathrm{InComp}\langle c, f', src, R', A', max, lead\rangle)\\
&\quad + \overline{b}(-, \mathbf{reply}, \langle \mathrm{id}, src\rangle, \mathrm{n}).\, \mathrm{InComp}\langle c, f', src, R', A', max, lead\rangle
\end{aligned}$$

$$\begin{aligned}
&\mathrm{Leader}\langle NULL, src, max, lead\rangle \stackrel{\text{def}}{=} \mathrm{Elected}\langle 0, src, max_i\rangle\\
&\mathrm{Leader}\langle f, src, max, lead\rangle \stackrel{\text{def}}{=}\\
&\quad r(\mathbf{election}, \langle j, l, s\rangle).\\
&\qquad \mathsf{cond}\ (l = lead \wedge s > src \rhd \mathrm{InComp}\langle 0, j, s, \mathrm{N} - \{j\}, \emptyset, max, lead\rangle,\\
&\qquad\quad true \rhd \mathrm{Leader}\langle f', src, max, lead\rangle)\\
&\quad + r(\mathbf{leader}, \langle l\rangle).\, \mathrm{Elected}\langle 0, src, l\rangle\\
&\quad + \overline{b}(-, \mathbf{reply}, \langle \mathrm{id}, src\rangle, \mathrm{n}).\, \mathrm{Leader}\langle f', scr, max, lead\rangle
\end{aligned}$$

Fig. 1. The description of a node

a set of connected components CC_g where independent leader-election computations are taking place within some subsets of nodes N_g. We prove the following:

Lemma 5. $CC_g \approx_s Spec_1$, *where* $Spec_1 \stackrel{\text{def}}{=} (\prod_{i \in N_g} \text{Elected}\langle 1, s_i, max_g\rangle{:}\sigma_i)\backslash T$ *and* $max_g = max\{max_i | i \in N_g\}$.

The proof takes place in two steps. In the first step, we consider a simplification of CC_g, F_g, where each node x in F_g is associated with a specific node being the unique node that x can accept as its father. We show that, by construction and Lemma 4, F_g is an S-confluent process and we exhibit an execution $F_g \Longrightarrow_s Spec_1$. Here we may observe the power of confluence: it is sufficient to study only a single execution of F_g. Then, by τ_s-inertness, S-confluence guarantees that $F_g \approx_s Spec_1$. For the second step of the proof we show that whenever $CC_g \Longrightarrow_s D$ where exists an F_g (i.e. an assignment of fathers to nodes) that is similar to D. Since this is true for any D we conclude that CC_g cannot diverge from the behavior of the F_g's and that it is in fact destined to produce only their behavior. Hence, we deduce that $CC_g \approx_s Spec_1$ as required.

Bearing in mind that network S_1 is a composition of the components CC_g, each concerning a distinct communication neighborhood of the network, S-confluence arguments along with considerations regarding the availability of believed maximum nodes allow us to deduce the correctness of the theorem. □

4 Conclusions

We have presented CSDT, a process calculus for reasoning about systems with broadcasting communication and dynamic interconnections. A salient aspect of CSDT is its ability to simulate a neighbor discovery protocol through its semantics and to associate nodes with their (believed) set of neighbors. Furthermore, we have allowed nodes to broadcast messages at two different transmission levels. As illustrated via our case study, the ability to do so is not only useful with respect to saving power but can also play an important role for protocol correctness. We point out that this could easily be extended to a wider range of transmission levels, by considering a set of transmission levels Tl and replacing sets N and H in a node's interface by a relation $\mathcal{I} \times Tl$. Finally, we have introduced the notion of a message type and a hiding operator based on types which allows an observer to focus on a subset of the message exchange. These message types are reminiscent of tags by which various applications prefix different messages according to their roles. As illustrated via our case study, this can be especially useful for expressing appropriate correctness criteria via bisimulations.

We have also studied the notion of confluence for CSDT in the presence and absence of node mobility. It turns out that in the case of stationary systems confluence yields a compositional theory which is remarkably simple compared to similar value-passing theories: the absence of channels for communication has removed a number of considerations relating to confluence preservation. We believe that the results can be extended to other calculi featuring a broadcasting style of communication. As in our case study, these results allow one to conclude the confluence of the analyzed systems merely by construction and then to deduce the desired bisimilarity via examining a single execution path and appealing to τ-inertness.

We have illustrated the applicability of the new formalism via the analysis of a leader-election MANET protocol. In [15], the authors also give a proof of their algorithm using temporal logic: in particular they show a "weak form of stabilization of

the algorithm", namely, that after a finite number of topological changes, the algorithm converges to a desired stable state in a finite amount of time. As we do in our proof, they operate under the assumption of no message loss. The same algorithm has also been considered in [13] where its correctness was analyzed for a number of tree and ring-structured initial topologies for networks with 5 to 8 nodes. It was shown automatically that it is possible that eventually a node with the maximum id in a connected component will be elected as the leader of the component and that every node connected to it via one or more hops will learn about its election. The reachability nature of this result is due to the lossy communication implemented in the ω-calculus.

In conclusion, we believe that CSDT can be applied for specifying and verifying a wide range of dynamic-topology protocols and that the theory of confluence can play an important role in facilitating the construction of their proofs. This belief is supported by our current work on specifying and verifying a MANET routing algorithm. In future work we plan to extend our framework in the presence of message loss.

References

1. F. Ghassemi, W. Fokkink, and A. Movaghar. Equational reasoning on mobile ad hoc networks. *Fundamenta Informaticae*, 105(4):375–415, 2010.
2. J. Ch. Godskesen. A calculus for mobile ad-hoc networks with static location binding. *Electronic Notes in Theoretical Computer Science*, 242(1):161–183, 2009.
3. D. Kouzapas and A. Philippou. A process calculus for systems with dynamic topology. Technical Report TR-10-01, University of Cyprus, 2010.
4. I. Lanese and D. Sangiorgi. An operational semantics for a calculus for wireless systems. *Theoretical Computer Science*, 158(19):1928–1948, 2010.
5. M. Merro. An observational theory for mobile ad hoc networks. *Information and Computation*, 207(2):194–208, 2009.
6. M. Merro and E. Sibilio. A timed calculus for wireless systems. In *Proceedings of FSEN'09, Revised Selected Papers*, LNCS 5961, pages 228–243, 2010.
7. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
8. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts 1 and 2. *Information and Computation*, 100:1–77, 1992.
9. S. Nanz and C. Hankin. A framework for security analysis of mobile wireless networks. *Theoretical Computer Science*, 367(1-2):203–227, 2006.
10. U. Nestmann. *On Determinacy and Non-determinacy in Concurrent Programming*. PhD thesis, University of Erlangen, 1996.
11. A. Philippou and D. Walker. On confluence in the π-calculus. In *Proceedings of ICALP'97*, LNCS 1256, pages 314–324, 1997.
12. K. V. S. Prasad. A calculus of broadcasting systems. *Science of Computer Programming*, 25(2-3):285–327, 1995.
13. A. Singh, C. R. Ramakrishnan, and S. A. Smolka. A process calculus for mobile ad hoc networks. In *Proceedings of COORDINATION'08*, LNCS 5052, pages 296–314, 2008.
14. C. Tofts. *Proof Methods and Pragmatics for Parallel Programming*. PhD thesis, University of Edinburgh, 1990.
15. S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Proceedings of ICNP'04*, pages 350–360. IEEE Computer Society, 2004.