

Restricted Broadcast Process Theory

Fatemeh Ghassemi

Sharif University of Technology, Tehran, Iran
fghassemi@mehr.sharif.edu

Wan Fokkink

Vrije Universiteit, Amsterdam, The Netherlands
wanf@cs.vu.nl

Ali Movaghar

Sharif University of Technology, Tehran, Iran
movaghar@sharif.edu

Abstract

We present a process algebra for modeling and reasoning about Mobile Ad hoc Networks (MANETs) and their protocols. In our algebra we model the essential modeling concepts of ad hoc networks, i.e. local broadcast, connectivity of nodes and connectivity changes. Connectivity and connectivity changes are modeled implicitly in the semantics, which results in a more compact state space. Our connectivity model supports unidirectional links. A key feature of our algebra is eliminating connectivity information from the specification of a network, and transferring its complexity to the semantics. We give a formal operational semantics for our process algebra, and define equivalence relations on protocols and networks. We show how our algebra can be applied to prove correctness of an ad hoc routing protocol.

1. Introduction

Mobile Ad hoc Networks (MANETs) are the ultimate frontier in wireless communication. They allow network nodes to communicate directly with each other using wireless transceivers (possibly along multihop paths) without the need for a fixed infrastructure. Ad hoc networks are expected to revolutionize wireless communications in the next few years; by exploiting ad hoc wireless technology, various portable devices (cellular phones, PDAs, laptops, pagers, and so on) and fixed equipment (base stations, wireless Internet access points, etc.) can be connected together, forming a sort of *global*, or *ubiquitous*, network [17].

The primitive means of communication in MANETs is *local broadcast*; only nodes located at the range of a transmitter receive data. Thus nodes participate in a broadcast according to the underlying topology of nodes; node A can receive data from node B , i.e. A is neighbor of B , if A is lo-

cated within the transmission range of node B . It should be noted that this neighbor relation is not symmetric or transitive. On the other hand, nodes move arbitrary, and the topology of network changes *dynamically*. Local broadcast, topology, and topology changes are the main modeling challenges in MANETs. Ad hoc protocols are defined irrespective of mobility behavior of nodes, so they should be verified against different mobility behaviors.

Protocols applied in ad hoc networks are usually tested by means of simulation. In simulation, a network of nodes is modeled and then run for a set of scenarios in a specific simulation environment. In each scenario, the set of events generated by nodes are specified. The simulation environment can take into account the physical area in which nodes are located, the time duration of simulation, the physical characteristics of nodes, and a node mobility model. A node mobility model defines the speed and movement direction of a node at each time. As the number of scenarios and environment simulation are restricted, by simulation one cannot explore all conditions that are required to hold for such systems. Formal methods can be used to model such networks, and then verify them using (semi-)automated model checking or theorem prover techniques.

Process algebras provide a good framework to reason about concurrent systems at the modeling level. We introduce a new process algebra to model concurrent behavior in the presence of topology changes. As explained in [5], from a theoretical point view, it appears difficult to encode broadcast in calculi based on point-to-point communications. Different process algebras have been proposed to address mobile wireless broadcast: $b\pi$ [6], CBS# [14], MBS [15], CWS [13], CMAN [7], CMN [12], and the ω -calculus [1], among which CBS#, CWS, CMAN, CMN, and the ω -calculus were deliberately tailored to address MANETs and their protocols. In all settings, the three above mentioned modeling challenges of ad hoc networks are addressed in different manners. In CWS, CMAN, CMN,

and the ω -calculus, the topology is defined as a part of the syntax, while the semantics of CBS# is quantified over a set of node configurations. In the former (except CWS, where the topology is considered static), a process evolves syntactically (to reflect topology changes) by the application of mobility rules defined in the semantics, while in the latter, the underlying configuration changes arbitrary in the semantics. Section 6, discusses related work in more detail.

In this paper we put forward a new value-passing process algebra, in which the topology changes dynamically in the semantics. We thus transfer topology concepts completely to the semantics (similar to CBS#), to let users focus on the specification. Transferring topology concepts to the semantics means each state represents a set of valid topologies, and a network can be at any of those topologies at any time. In addition, it is the network behavior that defines a set of valid topologies under which such behavior is correct, rather than that the underlying topology dictates the network behavior. A valid topology is a set of connectivity relations between nodes such that the *topology invariant* is satisfied. A topology invariant is a set of predicates defined over nodes. For instance, connections between nodes are always bidirectional, or there is always a path between nodes A and B . These predicates enable us to model ad hoc protocol assumptions and reduce the number of topologies that should be considered in a verification. This approach has some advantages:

- The process algebra specification of a MANET gives rise to a relatively small state space. In each state a set of topologies are valid. While we are in a state, nodes can move and make underlying topology switch between valid topologies. However, when we move from a state to another state, this transmission is valid under a set of topologies. This set is defined according to the network behavior, i.e. nodes that participated in the communication should be connected to the transmitter, while the connection between other nodes can be anything. In reality, after a broadcast communication, we can only find out the connectivity relation relative to the transmitter. Finding a set of topologies under which a network behavior can occur, is not amenable. Conversely, finding a set of topologies under which a network behavior occurred, is straightforward.
- According to [20], the correct operation of an ad hoc routing protocol is defined as follows: *If at one point in time there exists a path between two nodes, then the protocol must be able to find some path between the nodes. When a path has been found, and for the time it stays valid, it shall be possible to send packets along the path from the source node to the destination node.* We can examine this correctness, as will be explained in Section 5.

The core of our algebra is appropriate for the specification and verification of protocols above the *Multi Access Control* (MAC) layer. One could extend it with required concepts of layer protocols, to provide a suitable framework for the verification of those protocols. For instance [14] is a suitable framework for modeling security protocols, as it specifies a store for each node. The information stored describes the long-term commitment of nodes, which can be used in control flow analysis.

Structure of the paper. Section 2 provides an informal explanation about how we model basic concepts of ad hoc protocols. Section 3 presents the syntax and formal semantics of our algebra. Section 4 defines equivalence relations between protocols and networks. Section 5 illustrates an application of our algebra, and explains how it can be applied to reason about ad hoc routing protocols. Section 6 discusses related works. Finally, Section 7 contains our conclusions and directions for future work.

2. Modeling Ad Hoc Protocol Concepts

The main concepts we consider for designing a process algebra for ad hoc networks are local broadcast, connectivity and connectivity changes.

In the design of our algebra, we abstract away from services provided by the MAC and its layers beneath such as contention resolution. The abstraction model of the MAC layer affects the design of our algebra directly. If more than one channel is considered in the MAC layer, like the IEEE 802.11a protocol for WLAN computer communication, one needs to modify a channel-based process algebra like CWS, CMN or the ω -calculus. By contrast, we consider only one channel at the MAC layer, and at any time only one node can get the channel and broadcast.

Connectivity can be modeled explicitly as a part of the syntax. We avoid this approach, as we believe a more natural way to specify ad hoc networks is separation of connectivity description from network specification. Connectivity is modeled implicitly by a set of topologies.

There are two approaches in modeling topology changes; in one approach, topology changes are modeled explicitly in the specification, like [6], while in the other approach, topology changes are modeled implicitly in the semantics. The latter approach provides us with a natural way to model and verify ad hoc networks. Similarly there are two approaches to model topology changes implicitly in the semantics; one models mobility explicitly by defining a set of mobility rules, while the other models node mobility implicitly. As we explained in Section 1, we model topology changes implicitly in the semantics. To model node mobility implicitly, our semantics is parameterized by a topology invariant, which defines a set of possible topologies.

In our semantics, in each state, nodes can move implicitly, and change the underlying topology from one of the valid topologies to another. Thus, each state is quantified over a set of valid topologies. In explicit modeling of mobility in the semantics, each state is representative of a topology, and mobility is modeled by transitions between states, by the application of mobility rules. In our implicit modeling these states are bound together, which results in a relatively small state space. In Figure 1, the state spaces of both approaches are compared. As shown, in explicit modeling there are transitions between states with a unobservable action τ representing mobility behavior, while in the implicit modeling approach these are modeled implicitly in a state with a set of valid topologies (which is defined by a topology invariant). In our semantics, when a node broadcasts, only a subset of nodes receive the message. Consequently, we only realize connectivity between sender and receivers, while connectivity between other nodes can be anything. Thus a transition (behavior) is valid for a set of topologies, shown as a subscript for each transition.

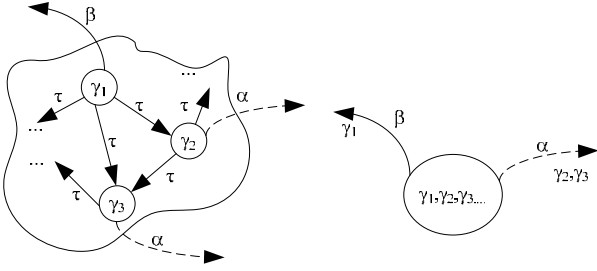


Figure 1. Comparison of the state spaces generated in explicit and implicit modeling of mobility in the semantics

3. Syntax and Semantics of Core Algebra

A network in our algebra consists of nodes composed in parallel. Some nodes may be hidden, and each node has a unique network address and deploys a process, called protocol. We first explain the syntax of our algebra and then its semantics by means of labeled transition systems.

3.1. Syntax

Let V denote a set of variables x, y , D a set of data values ranged over by u , and M a set of messages sent over a network and ranged over by m , together with a function $par : M \rightarrow \mathbb{N}$ to define the number of parameters encapsulated in a message. For instance, $m(x_1, \dots, x_k)$ denotes a message that encapsulates k parameters, with $m \in M$, $par(m) = k$ and $x_1, \dots, x_k \in V$.

Let \hat{x} denote a finite sequence of distinct variables x_1, \dots, x_k for some $k \in \mathbb{N}$, $|\hat{x}|$ its arity k , and $\{\hat{u}/\hat{x}\}$ the sequence of substitutions $\{u_1/x_1\}, \dots, \{u_k/x_k\}$.

Processes. We use the terms protocol and process interchangeably. The set of protocols is defined inductively by:

$$P ::= 0 \mid \alpha.P \mid P + P \mid [x = y]P, P \mid A(\hat{u}), A(\hat{x}) \stackrel{\text{def}}{=} P$$

0 specifies deadlock. $\alpha.P$ shows a process that is able to perform action α and then behaves as process P . The action α can be a send action $m(\hat{x})!$, or a receive action $m(\hat{x})?$ parameterized by m denoting the readiness of a protocol to send or receive a message of type m . The process $P_1 + P_2$ behaves non-deterministically as process P_1 or P_2 . The guarded command $[x = y]P_1, P_2$ (which is the same as *if-then-else*) defines process behavior based on $x = y$; if it evaluates to true, the protocol behaves as P_1 , and otherwise as P_2 . We write $A(\hat{u})$ to denote a process invocation defined via a recursive definition $A(\hat{x}) \stackrel{\text{def}}{=} P$, with $|\hat{x}| = |\hat{u}|$, where \hat{x} contains all names that appear free in P .

Names in u are not bound in P , by the use of α -conversion if needed. The sets of free and bound names in process P are denoted by $fn(P)$ and $bn(P)$ respectively. Parameters of message in $m(\hat{x})?.P$ are bound in P . Thus the set of free names is defined by: $fn(m(\hat{x})!.P) = \hat{x} \cup fn(P)$, and $fn(m(\hat{x})?.P) = fn(P) \setminus \hat{x}$, where \hat{x} is the set representation of \hat{x} .

Networks. Let L denote a set of location names, ranged over by l , which models the hardware addresses of nodes at which protocols run. The set of networks is defined by:

$$N ::= 0 \mid \llbracket P \rrbracket_l \mid (\nu l)N \mid N \parallel N$$

0 specifies a deadlock network. A protocol P at the location l is denoted by $\llbracket P \rrbracket_l$ and specifies a singleton node. $(\nu l)N$ binds the scope of l to the network N . The broadcasts from the node at l are hidden from an external observer. Finally, a network can be the parallel composition of processes running at nodes. The sets of free and bound locations in network N are denoted by $fl(N)$ and $bl(N)$, respectively. The set of free locations is defined inductively by $fl(\llbracket P \rrbracket_l) = \{l\}$, $fl((\nu l)N) = fl(N) \setminus \{l\}$, and $fl(N_1 \parallel N_2) = fl(N_1) \cup fl(N_2)$.

3.2. Labeled Transition System Semantics

We use structural operational semantics to define the formal semantics of our algebra. Given a protocol, the operational rules induce labeled transition systems in which transitions are $P \xrightarrow{\alpha} P'$, with $\alpha \in \{m(\hat{x})?, m(\hat{x})!\}$.

The formal semantics of processes is shown in Table 1. The Pre_1 and Pre_2 rules indicate execution of receive and send actions. After a process gets a message (of the same

type it was ready to receive), values of message parameters are replaced by the corresponding parameters. *Choice* specifies the choice operator; its symmetric version also holds, but is omitted here. Usually, ad hoc network protocols set timers to perform an action. We can use non-deterministic choice to model time-outs. Generally non-deterministic choice allows us to model alternative behaviors of a protocols against internal or external events. A guarded process behaves as P_1 if the guard condition evaluates to true (*Then*), otherwise it behaves as P_2 (*Else*). Process invocation is specified in the standard fashion in *Inv*, where $P\{\hat{u}/\hat{x}\}$ is obtained from P by replacing all free occurrences of variables x_i by u_i .

Table 1. Semantics of processes

$$\begin{array}{c}
\frac{}{m(\hat{x})?.P \xrightarrow{m(\hat{y})?} P\{\hat{y}/\hat{x}\}} : Pre_1 \\
\\
\frac{}{m(\hat{x})!.P \xrightarrow{m(\hat{x})!} P} : Pre_2 \qquad \frac{P_1 \xrightarrow{\alpha} P'_1}{P_1 + P_2 \xrightarrow{\alpha} P'_1} : Choice \\
\\
\frac{P_1 \xrightarrow{\alpha} P'_1}{[x = x]P_1, P_2 \xrightarrow{\alpha} P'_1} : Then \qquad \frac{P_2 \xrightarrow{\alpha} P'_2 \quad x \neq y}{[x = y]P_1, P_2 \xrightarrow{\alpha} P'_2} : Else \\
\\
\frac{P\{\hat{u}/\hat{x}\} \xrightarrow{\alpha} P' \quad A(x) \stackrel{\text{def}}{=} P}{A(\hat{u}) \xrightarrow{\alpha} P'} : Inv
\end{array}$$

A topology γ is a function defined over $L \rightarrow \mathbb{P}L$, where $\gamma(l)$ denotes the set of nodes connected to l .

Given a network of protocols, the operational rules in Table 2 induce a labeled transition system in which transitions are $N \xrightarrow{\beta}_{\Gamma} N'$, where Γ is a set of topologies, and β can be:

$$\beta ::= \tau \mid m(\hat{x})!\{l\}\nu g \mid m(\hat{x})?\sigma\nu g$$

where τ denotes an unobservable action, $m(\hat{x})!\{l\}\nu g$ denotes message m is broadcast from a node at location l and nodes at locations in g are hidden, and $m(\hat{x})?\sigma\nu g$ denotes the readiness of nodes at locations in σ to receive message m and nodes at locations in g are hidden. The only observable action is sending a message, because when a message is transmitted, we only know who has sent a message, but we don't know who has received it. As mentioned before, our semantics is parameterized with a topology invariant I that defines a set of valid topologies for each state (because of movements of nodes). If a topology γ satisfies topology invariant I , this is denoted by $\gamma \models I$. Each transition for networks carries as subscript a set of topologies Γ for which this transition is valid.

The operational semantics of networks is shown in Table 2. The symmetric versions of *Bro* and *Par* have been omitted. In this table, α is a send/receive action,

$\sigma, \sigma_1, \sigma_2, g, g_1, g_2 \subseteq L, N, N_1, N_2$ are networks, $\Gamma, \Gamma_1, \Gamma_2$ are sets of topologies, and P is a protocol. We write $(\nu\hat{g})N$ as an abbreviation for $(\nu l_1(\dots(\nu l_k)\dots))N$ where $l_i \in g, \sigma l$ for $\sigma \cup \{l\}$, $g_1 g_2$ for $g_1 \cup g_2$, and N for $(\nu\epsilon)N$. We require that no conflict occurs between free locations of a network. Rule *Inter* denotes that a single node performs an action under any valid topology, and its location (network address) is appended to the action. *Recv₁* defines a set of nodes ready to receive a message of type m . *Recv₂* hides receive actions to insist on their unobservability. *Bro* indicates the actual synchronization in local broadcast among transmitter and receivers. This transition is valid for all topologies in which the transmitter is connected (not essentially bi-directly) to the receivers defined by Γ_2 . The communication results in a transition labeled with $m(\hat{x})!\{l\}$, so the message $m(\hat{x})!$ remains visible to be received by other nodes. Hence we may have:

$$\begin{array}{l}
N_1 = \llbracket m(x)!.P_1 \rrbracket_A \xrightarrow{m(x)!\{A\}}_{\Gamma} \llbracket P_1 \rrbracket_A, \quad I = true \\
N_2 = \llbracket m(y)?.P_2 \rrbracket_B \parallel \llbracket m(z)?.P_3 \rrbracket_C \xrightarrow{m(x)!\{B,C\}}_{\Gamma} \\
\quad \llbracket P_2\{x/y\} \rrbracket_B \parallel \llbracket P_3\{x/z\} \rrbracket_C = N'_2 \\
N_1 \parallel N_2 \xrightarrow{m(x)!\{A\}}_{\Gamma'} \llbracket P_1 \rrbracket_A \parallel N'_2, \\
\quad \forall \gamma \in \Gamma'. \gamma \models \{B, C\} \in \gamma(A)
\end{array}$$

where $\{A, B, C, \dots\} \in L$. We note that Γ contains all possible topologies between nodes A, B and C (it includes 64 valid topologies). The last transition is valid for all topologies in which A is connected to both B and C . (Note that Γ' contains only 16 out of 64 valid topologies). The rules *Open*, *Close₁*, *Close₂*, *Close₃*, *Close₄* and *Close₅* make sure that extrusion of bound locations is treated properly. In *Close₁*, *Close₂*, *Close₃* and *Close₄* the send and receive actions of hidden nodes are concealed from an external observer, e.g.

$$\begin{array}{l}
N_3 = (\nu A)N_1 \parallel \llbracket m(y)?.P_4 \rrbracket_D \xrightarrow{m(x)!\{A\}\nu\{A\}}_{\Gamma} \\
\quad \llbracket P_1 \rrbracket_A \parallel \llbracket P_4\{x/y\} \rrbracket_D \\
N_2 = \llbracket m(y)?.P_2 \rrbracket_B \parallel \llbracket m(z)?.P_3 \rrbracket_C \xrightarrow{m(x)!\{B,C\}}_{\Gamma} \\
\quad \llbracket P_2\{x/y\} \rrbracket_B \parallel \llbracket P_3\{x/z\} \rrbracket_C = N'_2 \\
N_3 \parallel N_2 \xrightarrow{\tau}_{\Gamma''} (\nu A)(\llbracket P_1 \rrbracket_A \parallel \llbracket P_4\{x/y\} \rrbracket_D \parallel N'_2) \\
\quad \forall \gamma \in \Gamma''. \gamma \models \{B, C, D\} \in \gamma(A)
\end{array}$$

The semantics of broadcast communication is lossy, so some potential receivers may lose the message (even when they are connected to the transmitter and are ready to receive the message), some nodes can not receive a message of type m , and some nodes are disconnected from the transmitter. These nodes do not participate in the local broadcast, as explained in *Par*. For instance:

$$N_1 \parallel N_2 \xrightarrow{m(x)!\{A\}}_{\Gamma'''} \llbracket P_1 \rrbracket_A \parallel \llbracket P_2\{x/y\} \rrbracket_B \parallel \llbracket m(z)?.P_3 \rrbracket_C \\
\quad \forall \gamma \in \Gamma'''. \gamma \models \{B\} \in \gamma(A).$$

where node C may be disconnected from the transmitter, or

Table 2. Semantics of networks

$$\begin{array}{c}
\frac{P \xrightarrow{\alpha} P'}{\llbracket P \rrbracket_l \xrightarrow{\alpha\{l\}}_{\Gamma} \llbracket P' \rrbracket_l} : Inter, \quad \forall \gamma \in \Gamma. \gamma \models I \quad \frac{N \xrightarrow{\alpha\sigma\nu g}_{\Gamma} N'}{(\nu l)N \xrightarrow{\alpha\sigma\nu g l}_{\Gamma} N'} : Open \\
\\
\frac{N_1 \xrightarrow{m(\hat{x})?\sigma_1\nu g_1}_{\Gamma} N'_1 \quad N_2 \xrightarrow{m(\hat{x})?\sigma_2\nu g_2}_{\Gamma} N'_2}{N_1 \parallel N_2 \xrightarrow{m(\hat{x})?\sigma_1\sigma_2\nu g_1g_2}_{\Gamma} N'_1 \parallel N'_2} : Recv_1, \quad g_1 \cap fl(N_2) = g_2 \cap fl(N_1) = \emptyset \quad \frac{N \xrightarrow{m(\hat{x})?\sigma}_{\Gamma} N'}{N \xrightarrow{\tau}_{\Gamma} N'} : Recv_2 \\
\\
\frac{N_1 \xrightarrow{m(\hat{x})!\{l\}\nu g_1}_{\Gamma_1} N'_1 \quad N_2 \xrightarrow{m(\hat{x})?\sigma\nu g_2}_{\Gamma} N'_2}{N_1 \parallel N_2 \xrightarrow{m(\hat{x})!\{l\}\nu g_1g_2}_{\Gamma_2} N'_1 \parallel N'_2} : Bro, \quad \forall \gamma \in \Gamma_1. \gamma \models \sigma \in \gamma(l) \wedge \gamma \in \Gamma_2 \\
\quad g_1 \cap fl(N_2) = g_2 \cap fl(N_1) = \emptyset \\
\\
\frac{N \xrightarrow{m(\hat{x})!\{l\}\nu g}_{\Gamma} N'}{N \xrightarrow{m(\hat{x})!\{l\}}_{\Gamma} (\nu \hat{g})N'} : Close_1, \quad l \notin g \quad \frac{N \xrightarrow{m(\hat{x})!\{l\}\nu g}_{\Gamma} N'}{N \xrightarrow{\tau}_{\Gamma} (\nu \hat{g})N'} : Close_2, \quad l \in g \\
\\
\frac{N \xrightarrow{m(\hat{x})?\sigma\nu g}_{\Gamma} N'}{N \xrightarrow{m(\hat{x})?\sigma\setminus g}_{\Gamma} (\nu \hat{g})N'} : Close_3, \quad \sigma \setminus g \neq \emptyset \quad \frac{N \xrightarrow{m(\hat{x})?\sigma\nu g}_{\Gamma} N'}{N \xrightarrow{\tau}_{\Gamma} (\nu \hat{g})N'} : Close_4, \quad \sigma \subseteq g \\
\\
\frac{N \xrightarrow{\tau}_{\Gamma} N'}{(\nu l)N \xrightarrow{\tau}_{\Gamma} (\nu l)N'} : Close_5 \\
\\
\frac{N_1 \xrightarrow{\beta}_{\Gamma} N'_1}{N_1 \parallel N_2 \xrightarrow{\beta}_{\Gamma} N'_1 \parallel N_2} : Par, \quad fl(N_1) \cap fl(N_2) = bn(\beta) \cap fn(N_2) = \emptyset
\end{array}$$

may have lost the message, so the connection between A and C can be anything, and A is connected to B . This rule defines the semantics of broadcast communication as non-blocking output, indicating that when a protocol broadcasts a message to its peer, it does not block or wait for reception or even delivery of the message by its peer.

4. Network Bisimulation

In this section, we define an appropriate notion of simulation/bisimulation for ad hoc networks. Since some nodes are hidden from an observer, we define weak simulation/bisimulation to abstract away τ actions. As stated in Section 3.2, the labels of transitions, β , can be $m(\hat{x})!\{l\}\nu g$, $m(\hat{x})?\sigma\nu g$ and τ , but the only visible action to an external observer is the send action. By the application of $Close_1$, $Close_2$, $Close_3$, $Close_4$, $Close_5$, and $Recv_2$, the labels of transitions are $m(\hat{x})!\{l\}$ and τ , denoted by η . To define simulation/bisimulation, we introduce some notations:

- \Rightarrow_{Γ_k} denotes the reflexive and transitive closure of $\xrightarrow{\tau}_{\Gamma_i}$ where Γ_k is a set of topologies for which the k th transition is valid, i.e. $\Rightarrow_{\Gamma_k} = \xrightarrow{\tau}_{\Gamma_1} \dots \xrightarrow{\tau}_{\Gamma_i} \dots \xrightarrow{\tau}_{\Gamma_k}$.
- $\xRightarrow{\eta}_{\Gamma}$ denotes $\Rightarrow_{\Gamma_1} \xrightarrow{\eta}_{\Gamma_2} \Rightarrow_{\Gamma}$.

- $\xRightarrow{\hat{\eta}}_{\Gamma}$ denotes \Rightarrow_{Γ} if $\eta = \tau$ and $\xRightarrow{\eta}_{\Gamma}$ otherwise.

Definition 1 (Protocol Bisimilarity) A binary relation \mathcal{R}_p over protocols is a protocol simulation if $P_1 \mathcal{R}_p P_2$ implies: if $P_1 \xrightarrow{\alpha} P'_1$, then there is P'_2 such that $P_2 \xrightarrow{\alpha} P'_2$ and $P'_1 \mathcal{R}_p P'_2$. We say that P_2 simulates P_1 if there is a protocol simulation \mathcal{R}_p such that $P_1 \mathcal{R}_p P_2$. A relation \mathcal{R}_p is called protocol bisimulation if both \mathcal{R}_p and its converse are protocol simulations. We say that P_1 and P_2 are protocol bisimilar, written $P_1 \approx_p P_2$, if there is a protocol bisimulation \mathcal{R}_p such that $P_1 \mathcal{R}_p P_2$.

Definition 2 (Network Bisimilarity) A binary relation \mathcal{R}_n^I over networks is a network simulation if $N_1 \mathcal{R}_n^I N_2$ implies:

- $fl(N_1) = fl(N_2)$; and
- if $N_1 \xrightarrow{\eta}_{\Gamma_1} N'_1$ and $\Gamma_1 \models I$, then there is an N'_2 such that $N_2 \xRightarrow{\hat{\eta}}_{\Gamma_2} N'_2$, $\Gamma_2 \models I$ and $N'_1 \mathcal{R}_n^I N'_2$.

We say that N_2 simulates N_1 if there is a network simulation \mathcal{R}_n^I such that $N_1 \mathcal{R}_n^I N_2$. A relation \mathcal{R}_n^I is called network bisimulation if both \mathcal{R}_n^I and its converse are network simulations. We say that N_1 and N_2 are network bisimilar, written $N_1 \approx_n^I N_2$, if there is a network bisimulation \mathcal{R}_n^I such that $N_1 \mathcal{R}_n^I N_2$.

Protocol and network bisimilarities are equivalence relations over protocols and networks, respectively. We now sketch proofs that moreover they are congruence relations.

Theorem 1 *Let P_i and Q_i be protocols such that $P_i \approx_p Q_i$ for all $1 \leq i, j \leq \ell$, and $A_1, \dots, A_\ell \in L$, and R an arbitrary protocol. Then, for all topology invariants I :*

- $\alpha.P_i \approx_p \alpha.Q_i$.
- $P_i + R \approx_p Q_i + R$.
- $[x = y]P_i, R \approx [x = y]Q_i, R$ and $[x = y]R, P_i \approx [x = y]R, Q_i$
- $\llbracket P_i \rrbracket_{A_i} \approx_n^I \llbracket Q_i \rrbracket_{A_i}$.
- $(\nu l)\llbracket P_i \rrbracket_{A_i} \approx_n^I (\nu l)\llbracket Q_i \rrbracket_{A_i}$, for all locations l .
- $\llbracket P_1 \rrbracket_{A_1} \parallel \dots \parallel \llbracket P_\ell \rrbracket_{A_\ell} \approx_n^I \llbracket Q_1 \rrbracket_{A_1} \parallel \dots \parallel \llbracket Q_\ell \rrbracket_{A_\ell}$.

Proof. The proofs of the three first cases are straightforward. The proof for the fourth case is similar to the remaining cases, so we only sketch the proof for the fifth and sixth cases. We use some notations in our proofs, such as $\prod_{1 \leq i \leq k} P_i$ denotes $\llbracket P_1 \rrbracket_{A_1} \parallel \dots \parallel \llbracket P_k \rrbracket_{A_k}$, $\prod_{1 \leq i \leq l_1} P_i \prod_{l_1+1 \leq j \leq k} P_j$ denotes $\llbracket P_1 \rrbracket_{A_1} \parallel \dots \parallel \llbracket P_k \rrbracket_{A_k}$, and $\prod_{\ell} P_i$ denotes $\prod_{1 \leq i \leq \ell} P_i$.

Case 2. The proof proceeds by establishing that the relation $\mathcal{R}_n^I = \{((\nu l)\llbracket P_i \rrbracket_{A_i}, (\nu l)\llbracket Q_i \rrbracket_{A_i}) \mid P_i \approx_p Q_i\}$ is network bisimulation, where I is an arbitrary topology invariant. It is obvious that $fl((\nu l)\llbracket P_i \rrbracket_{A_i}) = fl((\nu l)\llbracket Q_i \rrbracket_{A_i})$. Suppose $(\nu l)\llbracket P_i \rrbracket_{A_i} \xrightarrow{\eta}_\Gamma (\nu l)\llbracket P'_i \rrbracket_{A_i}$. There are two possible cases for η .

- η is a send action such as $m(\hat{x})!\{A_i\}$, so by using rules *Close₁*, *Open*, *Inter* and *Pre₂*, we get $P_i \xrightarrow{m(\hat{x})!} P'_i$. As $P_i \approx_p Q_i$, there is a Q'_i such that $Q_i \xrightarrow{m(\hat{x})!} Q'_i$ and $P'_i \approx_p Q'_i$. By applying *Inter*, *Open* and *Close₁*, we get $(\nu l)\llbracket Q_i \rrbracket_{A_i} \xrightarrow{\eta}_\Gamma (\nu l)\llbracket Q'_i \rrbracket_{A_i}$ and $((\nu l)\llbracket P'_i \rrbracket_{A_i}, (\nu l)\llbracket Q'_i \rrbracket_{A_i}) \in \mathcal{R}_n^I$.
- η is a τ action. There are two possible cases. Let P_i perform a send action, and $A_i = l$. The proof is the same as previous one, but we should apply *Close₂* instead of *Close₁*. The other case is that P performs a receive action; we first consider $A_i \neq l$. Thus by the application of *Close₃*, *Open*, *Inter* and *Pre₁*, we have $P_i \xrightarrow{m(\hat{x})?} P'_i$; as $P_i \approx_p Q_i$, we have $Q_i \xrightarrow{m(\hat{x})?} Q'_i$ and then by applying *Inter*, *Open* and *Close₃*, we get $(\nu l)\llbracket Q_i \rrbracket_{A_i} \xrightarrow{\eta}_\Gamma (\nu l)\llbracket Q'_i \rrbracket_{A_i}$ and $((\nu l)\llbracket P'_i \rrbracket_{A_i}, (\nu l)\llbracket Q'_i \rrbracket_{A_i}) \in \mathcal{R}_n^I$ as $P'_i \approx_p Q'_i$. The proof when $A_i = l$ is the same, but we should use *Close₄* instead of *Close₃*.

Case 3. We need to show that $\mathcal{R}_n^I = \{(\prod_{\ell} P_i, \prod_{\ell} Q_i) \mid P_i \approx_p Q_i\}$ is a strong bisimulation, where I is an arbitrary topology invariant. It is obvious that $fl(\prod_{\ell} P_i) = fl(\prod_{\ell} Q_i)$. Suppose $\prod_{\ell} P_i \xrightarrow{\eta}_\Gamma N'_1$. There are two possible cases for η .

- η is a send action like $m(\hat{x})!\{A_i\}$ where $A_i \in \{A_1, \dots, A_\ell\}$. Suppose we have rearranged process and location subscripts such that P_1, \dots, P_{i-1} participate in local broadcast and P_{i+1}, \dots, P_ℓ do not participate. Thus by the application of *Par*, *Bro*, *Recv₁*, *Pre₁*, and *Pre₂* we derive the corresponding deduction tree as shown in Table 3 for the transition $\prod_{\ell} P_i \xrightarrow{m(\hat{x})!\{A_i\}}_\Gamma N'_1$, where N'_1 is $\prod_{1 \leq j \leq i} P'_j \prod_{i+1 \leq k \leq \ell} P_k$. The same tree can be derived for $\prod_{\ell} Q_i$ as $P_i \approx_p Q_i$. Consequently $\prod_{\ell} Q_i \xrightarrow{m(\hat{x})!\{A_i\}}_\Gamma N'_2$ such that $N'_2 = \prod_{1 \leq j \leq i} Q'_j \prod_{i+1 \leq k \leq \ell} Q_k$ and $(N'_1, N'_2) \in \mathcal{R}_n^I$.
- η is a τ action. By the application of *Recv₂*, there are a number of processes that perform a receive action. Similarly as above, we can derive the deduction tree for the $\prod_{\ell} P_i \xrightarrow{\tau}_\Gamma N'_1$ transition, by applying *Recv₂*, *Par*, *Recv₁* and *Pre₁*. The same tree can be derived for $\prod_{\ell} Q_i \xrightarrow{\tau}_\Gamma N'_2$ such that $(N'_1, N'_2) \in \mathcal{R}_n^I$.

Theorem 2 *Let N_1 and N_2 be two networks such that $N_1 \approx_n^I N_2$. Then,*

- $N_1 \parallel N \approx_n^I N_2 \parallel N$, for all networks N .
- $(\nu l)N_1 \approx_n^I (\nu l)N_2$, for all locations l .

Proof. We show that $\mathcal{R}_n^I = \{(N_1 \parallel N, N_2 \parallel N) \mid N_1 \approx_n^I N_2\}$ is a strong bisimulation. Suppose $N_1 \parallel N \xrightarrow{\eta}_\Gamma N'_1 \parallel N'$ where η is a send or τ action. We examine three cases:

- Suppose $N_1 \parallel N \xrightarrow{m(\hat{x})!\{l\}}_\Gamma N'_1 \parallel N'$ such that $N_1 \xrightarrow{m(\hat{x})!\{l\}}_\Gamma N'_1$. By the application of *Bro*, we can derive the following deduction tree:

$$\frac{N_1 \xrightarrow{m(\hat{x})!\{l\}}_\Gamma N'_1 \quad N \xrightarrow{m(\hat{x})?}_\Gamma N'}{N_1 \parallel N \xrightarrow{m(\hat{x})!\{l\}}_\Gamma N'_1 \parallel N'}$$

As $N_1 \approx_n^I N_2$, there is an N'_2 such that $N_2 \Rightarrow_{\Gamma'_3} \xrightarrow{m(\hat{x})!\{l\}}_{\Gamma'_2} \Rightarrow_{\Gamma'_1} N'_2$ with $N'_1 \approx_n^I N'_2$. By deriving the same deduction tree for the $\xrightarrow{m(\hat{x})!\{l\}}$ transition, and applying *Par* to \Rightarrow transitions, we get $N_2 \parallel N \xrightarrow{m(\hat{x})!\{l\}}_\Gamma N'_2 \parallel N'$ and $(N'_1 \parallel N', N'_2 \parallel N) \in \mathcal{R}_n^I$.

Table 3. Deduction tree corresponding to $\prod_{\ell} P_i \xrightarrow{m(\widehat{x})!\{A_i\}}_{\Gamma} N'_1$

$$\begin{array}{c}
 \frac{P_1 \xrightarrow{m(\widehat{x})?} P'_1 \quad \dots \quad P_j \xrightarrow{m(\widehat{x})?} P'_j}{\dots} \\
 \frac{[[P_1]]_{A_1} \xrightarrow{m(\widehat{x})?\{A_1\}}_{\Gamma_1} [[P_1\{x/y\}]]_{A_1} \quad [[P_j]]_{A_j} \xrightarrow{m(\widehat{x})?\{A_j\}}_{\Gamma_1} [[P_1\{x/z\}]]_{A_j}}{\dots} \\
 \frac{\prod_{1 \leq j \leq i-1} P_j \xrightarrow{m(\widehat{x})?\{A_1, \dots, A_{i-1}\}}_{\Gamma_1} \prod_{1 \leq j \leq i-1} P'_j \quad \dots \quad [[P_i]]_{A_i} \xrightarrow{m(\widehat{x})!\{A_i\}}_{\Gamma_2} [[P'_i]]_{A_i}}{\dots} \\
 \frac{\prod_{1 \leq j \leq i} P_j \xrightarrow{m(\widehat{x})!\{A_i\}}_{\Gamma} \prod_{1 \leq j \leq i} P'_j}{\dots} \\
 \frac{\prod_{\ell} P_i \xrightarrow{m(\widehat{x})!\{A_i\}}_{\Gamma} \prod_{1 \leq j \leq i} P'_j \prod_{i+1 \leq k \leq \ell} P_k = N'_1}{\dots}
 \end{array}$$

- Suppose $N_1 \parallel N \xrightarrow{m(\widehat{x})!\{l\}}_{\Gamma} N'_1 \parallel N'$ such that $N \xrightarrow{m(\widehat{x})!\{l\}}_{\Gamma_2} N'$. By application of *Bro*, there is a set of nodes in N_1 that participated in a local broadcast, i.e. $N_1 \xrightarrow{m(\widehat{x})?\sigma}_{\Gamma_1} N'_1$, and the nodes in σ are connected to l in Γ . As a receive action is an unobservable action, $N_1 \xrightarrow{\tau}_{\Gamma_1} N'_1$ and consequently there is an N'_2 such that $N_2 \Rightarrow_{\Gamma_1} N'_2$ and $N'_1 \approx_n^I N'_2$. By applying *Par* twice we derive :

$$\frac{\frac{N \xrightarrow{m(\widehat{x})!\{l\}}_{\Gamma_2} N'}{N_2 \parallel N \xrightarrow{m(\widehat{x})!\{l\}}_{\Gamma_2} N_2 \parallel N'} \quad \frac{N_2 \Rightarrow_{\Gamma_1} N'_2}{N_2 \parallel N' \Rightarrow_{\Gamma_1} N'_2 \parallel N'}}{N_2 \parallel N \xrightarrow{m(\widehat{x})!\{l\}}_{\Gamma_1} N'_2 \parallel N'}$$

and $(N'_1 \parallel N', N'_2 \parallel N') \in \mathcal{R}_n^I$. If no node in N_1 participates in the local broadcast, it can be easily proved by application of *Par*.

- Suppose $N_1 \parallel N \xrightarrow{\tau}_{\Gamma} N'_1 \parallel N'$. There are two cases:
 1. Suppose $N_1 \xrightarrow{\tau}_{\Gamma} N'_1$. Then by application of *Par*, we get $N_1 \parallel N \xrightarrow{\tau}_{\Gamma} N'_1 \parallel N$. As $N_1 \approx_n^I N_2$ there is an N'_2 such that $N_2 \Rightarrow_{\Gamma} N'_2$ and $N'_1 \approx_n^I N'_2$. By application of *Par* we get $N_2 \parallel N \Rightarrow_{\Gamma} N'_2 \parallel N$ too. Similarly, if $N \xrightarrow{\tau}_{\Gamma} N'$, it can be easily proved by application of *Par* in the same way.
 2. A local broadcast occurs from a hidden location in N_1 (or N). It can be proved in the same way as above when the transmitter node is not hidden, but we should add two steps: by applications of *Close₂* and *Open* to the deduction tree.

The following propositions are resulted from the definition of network bisimilarity:

Proposition 1 *Let N be a network and ℓ is a bound location in N , then for all topology invariants I , $N \approx_n^I (\nu \ell)N$.*

Proposition 2 *Let N be a network, then for any locations ℓ_1 and ℓ_2 , and for all topology invariants I , $(\nu \ell_1)(\nu \ell_2)N \approx_n^I (\nu \ell_2)(\nu \ell_1)N$.*

Proposition 3 *Let N_1 and N_2 be networks, then for all topology invariants I , $(\nu \ell)N_1 \parallel N_2 \approx_n^I (\nu \ell)(N_1 \parallel N_2)$ if $\ell \notin fl(N_2)$.*

5. A Case Study

In this section we go through a simple protocol, called SP in this context. This protocol is based on flooding a message, until the message reaches its destination. Nodes rebroadcast messages they have received. Thus if two nodes are not connected directly, they communicate via multiple intermediate nodes. We model *Initiator* who initiates a request, an *SP* who relays the message if it is not the target of the message, and otherwise behaves as *Destination* to send an acknowledgment. The specification of each process is as follows:

$$\begin{aligned}
 \text{Initiator}(\text{destip}, ip) &\stackrel{\text{def}}{=} Rreq(\text{destip}, ip)!. \\
 &(((Rrep(\text{srcip})?.\text{Initiator}(\text{dsetip}, ip)) + \\
 &\text{Initiator}(\text{destip}, ip)));
 \end{aligned}$$

$$\begin{aligned}
 SP(ip) &\stackrel{\text{def}}{=} Rreq(\text{target}, \text{sender})?. \\
 &[\text{target} = ip]\text{Destination}(\text{sender}), \\
 &(Rreq(\text{target}, \text{sender})!.SP(ip));
 \end{aligned}$$

$$\text{Destination}(\text{srcip}) \stackrel{\text{def}}{=} Rrep(\text{srcip})!. \text{Destination}(\text{srcip});$$

Here *Rreq* and *Rrep* are request and reply messages. The *Initiator* nondeterministically time-outs and retransmits its request to its desired destination. *SP* examines if it is the target of a request, otherwise it rebroadcasts the request. When a request reaches its destination, the destination node replies a number of times.

Consider a network that consists of three nodes A , B and C , where A deploys *Initiator*(B, A), while B and C deploy *SP*(B) and *SP*(C), respectively. We require that SP be correct, i.e. if at any time there is a path between A and B , A succeeds to send a message to B . In other words, a network in which there is a path between A and B behaves the same as a network where A and B are connected directly. So we choose the topol-

ogy invariant I as $connect(A, B)$ (where the predicate $connect$ is defined inductively by $connect(A, B) = B \in \gamma(A)$; $connect(A, B) = connect(A, C) \wedge connect(C, B)$), and show that having one (or even more) intermediate hidden nodes does not make any observational difference; i.e. $Spec_0 = \llbracket Initiator(B, A) \rrbracket_A \parallel \llbracket SP(B) \rrbracket_B \approx_n^I Imp_0 = \llbracket Initiator(B, A) \rrbracket_A \parallel (\nu C) \llbracket SP(C) \rrbracket_C \parallel \llbracket SP(B) \rrbracket_B$.

One can prove more about SP using model checking. We can derive scenarios in which a message does not reach its destination. Consider the topologies shown in Figure 2, where B is located near the transmission range of both A and C , so that it can connect to C while being disconnected from A , and vice versa. Similarly, C is located near the transmission range of A , so that it can repeatedly connect to and disconnect from A . In a scenario in which repeatedly B is connected to C when A broadcasts, and then connected to A when C broadcasts, while C repeatedly connects to and disconnects from A , B may not receive the message. This scenario is simulated by a recurrent transition of node A in the $Spec_0$ network.

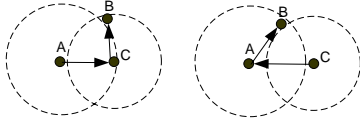


Figure 2. A scenario in which a message does not reach its destination

To be able to verify the correctness of SP, we should replace $Initiator$ with $Initiator_1$ in both $Spec_0$ and Imp_0 , where $Initiator_1(desipt, ip) \stackrel{\text{def}}{=} Rreq(dest, ip)!.Rrep(srcip)?.Initiator_1$. $Initiator_1$ abstracts from timeouts and retransmissions. Now it is easy to verify that $Spec_1 \not\approx_n^I Imp_1$. Generally speaking we can replace SP by any routing protocol to examine its correctness. It should be noted that according to the definition of correctness of ad hoc routing protocols explained in Section 1 (“When a path has been found, and for the time it stays valid, ...”), we should constrain I to verify its correctness against topology changes; I should constrain duration of changes in a path.

6. Related Work

The SPIN model checker has been applied in [4, 19, 2] to verify routing protocols in ad hoc networks, namely the Wireless Adaptive Routing Protocol (WARP) [10], the Lightweight Underlay Network Ad hoc Routing protocol [18] (LUNAR), and the Distance Vector Routing Protocols. Uppaal has been applied in [19, 20, 8, 11, 3] to verify real-time aspects of routing protocols in ad hoc networks.

The main properties considered for ad hoc network protocols in these works are *broadcast*, *connectivity* and *mobility* behavior of nodes. In both approaches, the connectivity of nodes is modeled by a two-dimensional array of boolean, and then mobility is modeled by manipulation of this matrix. Broadcast communication is handled by unicasting to all nodes with whom the sending node presently has connectivity, using the connectivity matrix. These approaches are not compositional. In [16], the backoff algorithm in ad hoc network is verified using PEPA, a stochastic process algebra. In this approach the topology of network is static and broadcast is implemented by unicasting.

Related calculi to ours are [14, 13, 7, 12, 1]. CBS# [14], an extension of CBS, models a node in a network by an $n[p, s]$ notation, specifying process p with store s deployed at the logical address of n . In this setting, similar to ours, only locations of nodes are specified. The semantics is parameterized by a topology, which consists of a set of connectivity graphs denoted by $\tau(I)$. A connectivity graph defines the invariant connections of a node in a network, and it belongs to a topology if it satisfies the topology invariant I . The neighbor relation in the connectivity graph is symmetric. The approach of CBS#, in the definition of topology and topology changes, is the same as ours, but our labeled transition systems and equivalence relations are completely different. In their approach, in any state, a transition to the next state is examined for all possible valid configurations, those satisfying the topology invariant, while in our approach a transition is examined for a subset of valid configurations (only the connections involved in a broadcast are examined). In CBS#, it is the configuration that defines the behavior of a network. By contrast, in our approach the behavior of a network defines the set of valid configurations (topologies) under which such a behavior can occur.

CWS [13] (Calculus for Wireless Systems) is a channel-based algebra for modeling of protocols at the data-link layer, at which interferences are an essential aspect. A node is specified by $n[p]_{l,r}^c$, denoting process p , synchronized on channel c , deployed at a node with logical address n , physical location l and transmission range r . The topology of a network is derived by a function d , defining which nodes are located within transmission range of each other. Locations of nodes cannot be changed. CMN [12] (Calculus of Mobile ad hoc Networks) is a value-passing calculus, inspired by CWS. Its node notation is the same as CWS (while c is called the mobility tag and denotes if a node is mobile or stationary), but with different semantics and definitions. Similar to CWS, a connection is derived by a function. The mobility rule in its semantics allows that nodes move and change their locations. This can cause problems for model checking, as the state space becomes infinite if locations are drawn from a real coordinate system.

In CMAN [7] (Calculus of Mobile Ad hoc Networks),

Table 4. Comparison between related algebras

	Node specification	Connectivity	Connectivity changes	Neighbor relation
CBS#	$n[p, s]$	$\tau(I)$	implicit-implicit	symmetric
CWS	$n[p]_{l,r}^c$	$d(l, l')$	static	symmetric
CMN	$n[p]_{l,r}^c$	$d(l, l')$	implicit-explicit	symmetric
CMAN	$[p]_l^\sigma$	σ	implicit-explicit	symmetric
ω -calculus	$p : g$	g	implicit-explicit	symmetric
RBPT (ours)	$\llbracket p \rrbracket_\ell$	I	implicit-implicit	asymmetric

the topology is a part of the syntax and can be changed using mobility rules in semantics. A node is specified by $[p]_l^\sigma$, denoting process p located at l and connected to the set of nodes defined by σ . The connection between nodes is symmetric and changes symmetrically by removing (or adding) a moving node’s location from (to) its previous (new) neighbors. This explicit handling of connection information affects the modularity of the semantics (the definition of bisimulation, in particular), and may preclude reasoning about open systems.

The ω -calculus [1], a conservative extension of the π -calculus, separates a node’s computational behavior described by an ω -process from the description of its physical transmission range, referred to as an ω -process interface. A node is defined by $p : g$, specifying process p connected to groups defined in g . A group is a maximal clique in a connectivity graph. Two nodes communicate if they belong to a same group. Consequently the connections in this model are considered bidirectional. Node mobility is captured through the dynamic creation of new groups and dynamically changing process interfaces, using appropriate rules in the semantics. By defining a set of mobility invariants which constraints node mobilities, one can verify a model against mobility scenarios.

In Table 4, we have compared our algebra with the related ones in terms of node specification, connectivity, connectivity changes (based on categories explained in Section 2), and neighbor relation between nodes. Among these approaches, only in CBS# and ours, topology is not specified as a part of the syntax, which allows to naturally specify and then verify an ad hoc network of arbitrary size with a dynamic underlying topology. The set of connectivities against which verification is performed is managed by topology invariants in CBS#, ω -calculus and ours. Topology invariants provide a flexible framework in verification. Our algebra and CBS# model connectivity changes implicitly in the semantics, which results in a relatively small labeled transition system, and as explained above our approach is even more compact compared to CBS#. As some of the protocols assume asymmetric links between nodes, only our approach supports modeling asymmetric relations between nodes. CBS# provides a framework for security analysis of ad hoc protocols based on control flow analy-

sis, while our approach provides analysis of concurrent processes based on protocol and network bisimilarity.

7. Conclusions and Future Work

In this paper, we have introduced a new process algebra to specify and verify MANETs and their protocols. To design our algebra, we considered main modeling challenges such as non-blocking and lossy local broadcast, connectivity, and connectivity changes. We model connectivity implicitly by a topology invariant, which assigns a set of valid topologies to each state. Connectivity changes are also modeled implicitly in the semantics, by allowing arbitrary switches between valid topologies in each state. Transferring the connectivity model and consequently its description completely to the semantics, provides a natural process algebra to model MANETs and their protocols. A transition between two states is defined under a set of topologies called Γ , meaning that a network behaves accordingly if its current topology is included in Γ . Definition of Γ leads to a more compact labeled transition system, and enables us to verify some properties regarding the underlying topology, for ad hoc routing protocols.

We gave an operational semantics, and defined equivalence notions between protocols and networks. We can in principle analyze a network process in our algebra by model checking, and examine the behavior of the network when Γ changes. We can also perform scenario-based model checking on protocols, by imposing some scheduling on Γ .

We may need to freeze for some duration the underlying topologies, to verify reconfigurable behavior of protocols. For instance, after a link breakage in a topology, we should fix the underlying topologies to let routing protocols deployed at nodes find the correct routes.

We plan to find a sound and complete axiomatization w.r.t. protocol and network bisimulations, to reason about networks without the need of generating their state spaces. We are going to extend our algebra to verify security properties of ad hoc networks based on information flow analysis [9]. We are also going to examine how to prove a property for networks of arbitrary size, using *Data Independent Structural Induction*. We also plan to prepare an automated

tool using Prolog, to verify networks and protocols.

References

- [1] S. A. S. Anu Singh, C. R. Ramakrishnan. A process calculus for mobile ad hoc networks. In *Proc. 10th International Conference on Coordination Models and Languages (COORDINATION)*, 2008. To appear.
- [2] K. Bhargavan, D. Obradovic, and C. A. Gunter. Formal verification of standards for distance vector routing protocols. *Journal of the ACM*, 49(4):538–576, 2002.
- [3] S. Chiyangwa and M. Kwiatkowska. A timing analysis of AODV. In *Proc. 7th IFIP International Conference on Formal Methods for Open Object-based Distributed Systems*, volume 3535 of *LNCS*, pages 306–321. Springer, 2005.
- [4] R. de Renesse and A. Aghvami. Formal verification of ad-hoc routing protocols using SPIN model checker. In *Proc. 12th IEEE Mediterranean Electrotechnical Conference*, pages 1177–1182. IEEE, 2004.
- [5] C. Ene and T. Muntean. Expressiveness of point-to-point versus broadcast communications. In *Fundamentals of Computation Theory*, pages 258–268, 1999.
- [6] C. Ene and T. Muntean. A broadcast-based calculus for communicating systems. In *IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium*, page 149, Washington, DC, USA, 2001. IEEE.
- [7] J. C. Godskesen. A calculus for mobile ad hoc networks. In A. L. Murphy and J. Vitek, editors, *COORDINATION*, volume 4467 of *Lecture Notes in Computer Science*, pages 132–150. Springer, 2007.
- [8] J. C. Godskesen and O. Gryn. Modeling and verification of security protocols for ad hoc networks using UPPAAL. In *Proc. 18th Nordic Workshop on Programming Theory*, 2006.
- [9] D. P. Gruska. Information flow in networks. In L. Czaja, editor, *Proceedings of the International Workshop on Concurrency, Specification and Programming (CS&P'05)*, pages 184–196. Warsaw University, 2005.
- [10] P. Khengar and A. Aghvami. Warp - the wireless adaptive routing protocol. In *Proc. IST Mobile Communications Summit 2001*, pages 480–485, 2001.
- [11] A. McIver and A. Fehnker. Formal techniques for analysis of wireless network. In T. Margaria, A. Philippou, and B. Steffen, editors, *Proc. 2nd IEEE-EASST International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, pages 263–270. IEEE, 2006.
- [12] M. Merro. An observational theory for mobile ad hoc networks. *Electr. Notes Theor. Comput. Sci.*, 173:275–293, 2007.
- [13] N. Mezzetti and D. Sangiorgi. Towards a calculus for wireless systems. *Electr. Notes Theor. Comput. Sci.*, 158:331–353, 2006.
- [14] S. Nanz and C. Hankin. A framework for security analysis of mobile wireless networks. *Theor. Comput. Sci.*, 367(1):203–227, 2006.
- [15] K. V. S. Prasad. A prospectus for mobile broadcasting systems. *Electr. Notes Theor. Comput. Sci.*, 162:295–300, 2006.
- [16] T. Razafindralambo and F. Valois. Performance evaluation of backoff algorithms in 802.11 ad-hoc networks. In *Proc. International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 82–89. ACM, 2006.
- [17] P. Santi. *Topology Control in Wireless ad Hoc and Sensor Networks*. John Wiley, 2005.
- [18] C. Tschudin, R. Gold, O. Rensfelt, and O. Wibling. LUNAR: a lightweight underlay network ad-hoc routing protocol and implementation. In *Next Generation Teletraffic and Wired/Wireless Advanced Networking*, page 300, 2004.
- [19] O. Wibling, J. Parrow, and A. Pears. Automated verification of ad hoc routing protocols. In *Proc. 24th IFIP WG6.1 International Conference on Formal Techniques for Networked and Distributed Systems*, volume 3235 of *LNCS*, pages 343–358. Springer, 2004.
- [20] O. Wibling, J. Parrow, and A. Pears. Ad hoc routing protocol verification through broadcast abstraction. In *Proc. 25th IFIP WG6.1 International Conference on Formal Techniques for Networked and Distributed Systems*, 2005.

A Appendix

To show that $Spec_0 \approx_n^I Imp_0$ holds, we need to find a bisimulation relation \mathcal{R}_n^I such that $(Spec_0, Imp_0) \in \mathcal{R}_n^I$. We represent $Rrep(destip, ip)?.Initiator$ with $Initiator'$ and $Rreq(target, sender)!.SP(ip)$ with SP' . For the sake of presentation, we abstract away from parameters in the following specifications. There are four states in $Spec_0$ represented by A, B, C and D as follows:

$$\begin{aligned} A &: \llbracket Initiator \rrbracket_A \parallel \llbracket SP \rrbracket_B \\ B &: \llbracket Initiator' \rrbracket_A \parallel \llbracket SP \rrbracket_B \\ C &: \llbracket Initiator' \rrbracket_A \parallel \llbracket Destination \rrbracket_B \\ D &: \llbracket Initiator \rrbracket_A \parallel \llbracket Destination \rrbracket_B \end{aligned}$$

We partition the states of Imp_0 into four groups called A', B', C' and D' ; the members of each group are bisimilar to states A, B, C and D , respectively.

$$\begin{aligned} A' &: \llbracket Initiator \rrbracket_A \parallel (\nu C) \llbracket SP \rrbracket_C \parallel \llbracket SP \rrbracket_B \\ B' &: \llbracket Initiator' \rrbracket_A \parallel (\nu C) \llbracket SP' \rrbracket_C \parallel \llbracket SP \rrbracket_B \\ & \llbracket Initiator' \rrbracket_A \parallel (\nu C) \llbracket SP' \rrbracket_C \parallel \llbracket Destination \rrbracket_B \\ & \llbracket Initiator' \rrbracket_A \parallel (\nu C) \llbracket SP \rrbracket_C \parallel \llbracket Destination \rrbracket_B \\ C' &: \llbracket Initiator \rrbracket_A \parallel (\nu C) \llbracket SP' \rrbracket_C \parallel \llbracket SP \rrbracket_B \\ & \llbracket Initiator \rrbracket_A \parallel (\nu C) \llbracket SP' \rrbracket_C \parallel \llbracket Destination \rrbracket_B \\ & \llbracket Initiator \rrbracket_A \parallel (\nu C) \llbracket SP \rrbracket_C \parallel \llbracket Destination \rrbracket_B \\ D' &: \llbracket Initiator' \rrbracket_A \parallel (\nu C) \llbracket SP \rrbracket_C \parallel \llbracket SP \rrbracket_B \end{aligned}$$

Thus if we show the members of groups A', B', C' and D' with a', b', c' and d' , respectively, then our bisimulation relation is:

$$\begin{aligned} \mathcal{R}_n^I = \{ & (a', A), (b', B), (c', C), (d', D) \} \\ & \forall a' \in A, \forall b' \in B, \forall c' \in C, \forall d' \in D \} \end{aligned}$$