The CSI Journal on Computer Science & Engineering

The CSI Journal on
Computer Science and Engineering
Vol. 5, No. 2&4 (b), 2007
Pages 46-55
Regular Paper

Formal Modeling Routing Protocols in Mobile Ad Hoc Networks

Fatemeh Ghassemi Ali Movaghar

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Abstract

In this paper we provide a process algebra to formally analyze ad hoc protocols above network layer. Mainly our focus in this paper is on formal modeling and analysis of routing protocols. We explain the semantics of our algebra using labeled transition systems and then define an equivalence relation between them. Finally we illustrate application of our algebra by a case study on cluster formation process of cluster based ad hoc routing protocol.

Keywords: Ad hoc Network Algebra, Restricted Broadcast, Topology Changes.

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 without the need for a fixed infrastructure. The protocols applied in such networks are usually tested by means of simulation. As is well-known, by simulation one cannot explore all conditions that are required to verify such systems. On the other hand, formal methods can be used to model such protocols, and then verify them using (semi) automated model checking or theorem prover techniques. In this paper first we explain different formal approaches applied to verify protocols at the network layer, i.e. routing protocols in ad hoc networks and then we continue with formal methods available for verification of network protocols (not essentially ad hoc) and finally explain algebras related to our calculi.

Process algebras provide strong techniques to reason about these systems at the modeling level. Process algebras are basically used to model communicating and concurrent systems such as Communicating Sequential Processes, CSP [1], Calculus of Communicating Systems, CCS [2], and Algebra of Communicating Processes, ACP [3]. Pi-calculus [4] and its extensions allow modeling distributed and mobile systems. In these algebras, communications, either inside the system or between the system and its environment, are modeled by synchronous actions on shared channels. The topology of the network is often assumed to be static, and detailed information about messages and their structure is not considered. The topology of ad hoc networks however has a direct impact on the behavior of routing protocols. We introduce a new process algebra to verify concurrency behaviors of processes in the presence of topology changes. Connectivity of a network is modeled by a connectivity function and is embedded implicitly in the semantics.

Related works. The Spin model checker has been applied in [5,6,7] to verify routing protocols in ad hoc networks, namely Wireless Adaptive Routing Protocol (WARP) [8], Lightweight Underlay Network Ad hoc Routing protocol (LUNAR) [9], and Distance Vector Routing Protocols. Uppaal has been applied in [6,11-14] to verify real-time aspects of routing protocols in ad hoc networks as a network of timed automata. 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 booleans, 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. In [14], 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.

There are some related works on modeling network protocols. The Calculus of Broadcasting Systems, CBS [15], is a process algebra in which the only means of communication is broadcast. When a process broadcasts, all other processes receive. We modify global broadcast as defined in CBS to broadcast at wireless network (local broadcast). Nomadic pi [16], an extension of pi-calculus, is suitable for verifying distributed and mobile systems. This algebra provides a two-level framework; at the lower level location-dependent primitives are introduced, while at the second layer location-independent operations are defined using the underlying primitives. This algebra can model distributed systems appropriately (at the application layer). In Nomadic pi, communication is channel-based; a process transmits data on a channel and another process receives data from the channel synchronously. In [17] a process algebra for network communication is introduced which focuses on real-time and capacity constraints of channels. In [18] pi-calculus is extended by a set of specific patterns for expressing protocol behavior such as route discovery and explicit congestion notification. However, this approach does not consider broadcast and topology changes. In [19] an algebraic model of an ad hoc network for mobile computing is developed. This model is achieved by the interplay of two algebras; the pathset algebra Z which provides a means to model the infrastructure, and a modal algebra for modeling multi agent systems. This approach provides a unified framework for modeling application layer protocols considering the underlying infrastructure. This approach is appropriate for modeling a system where interactions at the application layer affect the underlying structure.

Related calculi to ours are [20-25]. CBS# [20], an extension of CBS, models a node by considering a store to maintain data received. This framework is more suited to verify security ad hoc protocols and is based on flow analysis. The mobility is modeled in semantics by assigning a set of topologies to each state and defining a behavior for each topology. This work is more similar to our earlier work in [25] with difference that in ours a behavior is defined for a set of topologies. CWS [21] (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. CMN [23] (Calculus of Mobile ad hoc Networks) is a value-passing calculus, inspired by CWS which models mobility by mobility rules in semantics. In CMAN [22] (Calculus of Mobile Ad hoc Networks), the topology is a part of the syntax and are explicitly handled in semantics which affects the modularity of the semantics. The ω-calculus [24], a conservative extension of the π-calculus, separates a node's computational behavior from the description of its physical transmission range. A complete comparison between above mentioned algebras can be found in [25]. They are compared in terms of their specification and modeling concepts.

This paper is based on earlier work in [26] and is completely different from works in [25,27]; we have modeled mobility in this paper using pre-condition concepts in semantics which are predicates over connectivity function, while in [25], we have modeled it by computing the set of possible topologies in semantics. Computing the set of topologies is not easy and cannot be automated in verifying an application easily. Although in [27], we have almost followed a same line on modeling mobility, these works based on different technicalities to follow different aims at mind. In [27], we focused on deriving an equational system, and based on an extension of branching bisimilarity called *branching computed network bisimilarity*. However in this paper, we aim at modeling true concurrency (not interleaving semantics) of send actions occurring in different parts of a network and the same as [25], is based on an extension of observational congruency called *network bisimilarity*. Besides with difference aims, there are some differences on syntax (different operators and actions).

The rest of the paper is organized as follows: in section 2 we explain our algebra, its syntax and semantics. In Sections 3, we define equivalence relations on protocols and networks. Section 4 presents a case study on the cluster formation process of cluster-based routing protocol. Section 5 contains our conclusions and directions for future work.

2. Routing Process Algebra

As mentioned in Section 1, in this paper we focus on modeling and verifying protocols at the network layer in ad hoc networks. As most of the protocols at this layer are routing protocols, we call this process algebra Routing Process Algebra (RPA). We abstract from MAC layer by assuming interference range and decoding range of each node are equivalent and the contention between nodes has been resolved as explained in Section 2.2.

The properties we consider for ad hoc protocols are local broadcast, connectivity and topology changes. When a message is sent by a node, only the nodes located in the range of the transmitter will receive the message. Our algebra should therefore support local broadcast. We have modified the global broadcast operation of CBS [15] to local broadcast; only adjacent nodes to the sender receive the broadcasted message. Ad hoc protocols should behave correctly according to the underlying topology and topology changes (dynamic). Connectivity is modeled by a connectivity function denoted by Γ. As the mobility behavior of nodes is not specified in a protocol specification, it should be implemented implicitly to verify the protocol against topology changes. We have implemented mobility in the semantics by imposing some restriction over connectivity function.

2.1. RPA Syntax

The syntax of RPA is shown below:

$N ::= [\![P]\!]_{A_1} \parallel \ldots \parallel [\![P]\!]_{A_n}$ $A_1,\ldots,A_n \in Names$

$P ::= 0 \mid 1 \mid a.P \mid P+P \mid P \bowtie_s P \mid [B]P \mid P/h$,
where $a \in Actions$, $s,h \subseteq Actions$

$B ::= B \wedge B \mid B \vee B \mid \neg B \mid R$

$R ::= R=R \mid R \leq R \mid R \geq R \mid R<R \mid R>R \mid R \neq R \mid x \mid d$,
where $x \in Variables$, $d \in Data$

where 0 specifies a deadlock (unsuccessful termination). In contrast, 1 specifies a successful termination. $a.P$ shows a process that is able to perform action a and then behaves like process P. The action a in the set *Actions* can be a send action *mx*! or *md*! (with *d* a datum from *Data*) or a receive action *mx*? or *md*? (with *x* variable in *Variables*) used in a

broadcast communication where m is a predefined message type m in *Message* set, or an internal action at one of the processes. The occurrence of variable x in process $mx!.P$ is free while it is bound in process $mx?.P$. The only binding operator in RPA is "receive" action. The process P_1+P_2 behaves non-deterministically like process P_1 or P_2. The process $P_1 \bowtie_s P_2$ defines CSP-like multi-way synchronization of two processes P_1 and P_2 on a set s of actions. The guarded command enables one to influence process behavior based on data received via a message transfer. For example, the process $[x>10]P$ behaves like P if the value of x is greater than 10, and like 0 otherwise. To encapsulate some internal behavior of a process, hide operator is used. The process P/h insures the action defined in h cannot be synchronized with any other processes *Names* is a set of node names, which model hardware addresses of the nodes at which processes can run. A process P at a node A_i is denoted by $[\![P]\!]_{A_i}$. A network, $[\![P]\!]_{A_1} \| \ldots \| [\![P]\!]_{A_n}$, is defined by the composition of processes $P_1,\ldots,P_n$ which run at nodes $A_1,\ldots,A_n$, via broadcast communication.

A network or process term is closed if the set of its free variables are empty and otherwise it is open. The set of bound variables of a network is defined as given in Table 1, where b is an internal action, not send and receive actions. The set of free variables are defined similarly.

Broadcast communication composition can only be defined between processes running at different nodes. The inner structures of processes at nodes can be sequence ".", choice "+", multi-way synchronization "⋈", guarded command "[]", and hide operator "/". The set of local processes that can be deployed at a node is represented by *Protocol*, and the set of broadcast communicating processes, composed of protocols using broadcast communication operator, is represented by *Network*. Physical locations in ad hoc networks are modeled by set of *Names*. The connectivity, network topology, is defined by the function Γ:*Names*→ℙ*Names*, and defines the immediate neighbors of each node in the network. This function need not be symmetric; possibly node A is connected to node B, while at that time B is not (yet) connected to A.

We add recursion to our syntax to support infinite computation. A process declaration consists of a collection E of recursive equations $X_1=P_1,\ldots, X_n=P_n$, where $P_1,\ldots,P_n$ can contain occurrences of the recursion variables $X_1,\ldots,X_n$. We write $\langle X_i|E\rangle$ (with i=1,…,n) for the process that behaves as X_i, with respect to E.

2.2. RPA Semantics

We use structural operational semantics to define the formal semantics of RPA operations. Each semantic rule consists of two parts: pre-condition (at the left) and operational rule. Pre-conditions are specified by prepositions on topology (modeled by connectivity function). The operational rules are defined on closed terms.

Table 1. Bound variables in a proves/network term

bv($mx!.P$)=bv(P)\{x}	bv($mx?.P$)=bv(P) ∪ {x}
bv($md!.P$)=bv(P)	bv($md?.P$)=bv(P)
bv($b.P$)=bv(P)	bv(P_1+P_2)=bv(P_1) ∪ bv(P_2)
bv($[x=d]P_1,P_2$)= bv(P_1)∪ bv(P_2)\ {x}	bv($[d=x]P_1,P_2$)=bv(P_1) ∪ bv(P_2)\ {x}
bv($[x=y]P_1,P_2$)=bv(P_1) ∪bv(P_2)\{x,y}	bv($\langle X\|E\rangle$)=bv(E)
bv($P_1 \bowtie_s P_2$)= bv(P_1) ∪ bv(P_2)	bv(P/h)=bv(P)
bv($[\![P]\!]_{A_i}$)=bv(P)	bv($N_1 \| N_2$)= bv(N_1) ∪ bv(N_2)

Table 2. Semantics of RPA

$$B = true \frac{P\downarrow}{[B]P\downarrow} : Gau_1 \qquad \frac{}{1\downarrow} : Succ \qquad \frac{}{a.P \xrightarrow{a} P} : Succ$$

$$B = true \frac{P \xrightarrow{a} P'}{[B]P \xrightarrow{a} P'} : Gau_2 \qquad \frac{P_1\downarrow}{P_1+P_2\downarrow} : Choice_1 \qquad \frac{P_1 \xrightarrow{a} P_1'}{P_1+P_2 \xrightarrow{a} P_1'} : Choice_2$$

$$\frac{\langle P_X \mid E\rangle\downarrow}{\langle X \mid E\rangle\downarrow} : Rec_1 \qquad \frac{\langle P_X \mid E\rangle \xrightarrow{a} Q}{\langle X \mid E\rangle \xrightarrow{a} Q} : Rec_2 \qquad a \in S, \frac{P_1 \xrightarrow{a} P_1' \;\; P_2 \xrightarrow{a} P_2'}{P_1 \triangleright\!\triangleleft_s P_2 \xrightarrow{a} P_1' \triangleright\!\triangleleft_s P_2'} : Comp_2$$

$$\frac{P_1\downarrow \; P_2\downarrow}{P_1 \triangleright\!\triangleleft_s P_2\downarrow} : Comp_1 \qquad a \notin S, \frac{P_1 \xrightarrow{a} P_1'}{P_1 \triangleright\!\triangleleft_s P_2 \xrightarrow{a} P_1' \triangleright\!\triangleleft_s P_2} : Comp_3$$

Given a protocol, the operational rules of RPA induce a labeled transition system in which transitions are $P \xrightarrow{a} P'$, with $a \in Actions$. The predicate ↓ denotes successful termination of a protocol.

The formal semantics of sequence, choice, hide and multi-way synchronization are straightforward as shown in Table 2. The *Succ* rule indicates termination. $Choice_1$ and $Choice_2$ specify the choice operator. In multi-way

synchronization, processes are synchronized on actions inside the synchronization set s ($Comp_1$ and $Comp_2$). In $Comp_2$, P_1 and P_2 perform the same action a. Actions outside the synchronization set s have the interleaving property, meaning that the process arguments can perform such actions independently ($Comp_3$). A guarded process can behave as the process in the argument, if the guard condition evaluates to true (Gau_1 and Gau_2). Recursion is specified in the standard fashion in Rec_1 and Rec_2; in these rules, P_X denotes the right-hand side of the equation for X in E, and $\langle P_X | E \rangle$ is obtained from P_X by replacing each recursion variable Y by $\langle Y|E \rangle$. The symmetric versions of $Choice_2$ and $Comp_3$, for the second argument of choice and multi-way synchronization, respectively, have been omitted.

Given a network of protocols, the operational rules of RPA induce a labeled transition system in which transitions are $N \xrightarrow{\alpha} N'$, where α is either τ representing an internal action, or β which is a set of terms like $md![A]$ denoting that a datum d is broadcast from node A, or $md?\sigma$ where $\sigma \subseteq Names$ denoting a set of nodes at locations defined by σ can receive the message $md?$. As before, $\downarrow$ denotes successful termination of a network. For example the set $\{m_1d_1[A_1], m_1d_2[A_2]\}$ denotes nodes at locations A_1 and A_2 where $A_1 \neq A_2$ are sending messages m_1d_1 and m_2d_2 respectively. Each transition has a set of pre-conditions resulted from the conjunction of all pre-conditions of SOS rules involved in deduction of that transition.

The operational semantics for broadcast communication is shown in Table 3. In this table the symmetric version of rules Bro_4, Par_1 and Par_2 has been removed. Bro_1 expresses that when a process broadcasts a message in the network, it appends its address to the message. Bro_2 explains when a node can receive a message; its name is added to the list of receivers of that message. We have merged these two cases; receiving an arbitrary message like $mx?$ or receiving a specific message $md?$; the second case has been shown in parenthesis. Bro_3 indicates a set of nodes can be synchronized by each other in receiving a message. Bro_4 indicates that a set of nodes specified

in σ can participate in communication with a sender if receivers were connected before or they have just connected to the sender indicated by pre-condition $\sigma \in \Gamma(A)$ (we will explain it more in Section 2.3). The communication results in a transition labeled with $\{md![A]\}$, so the message m remains visible for other nodes awaiting message m. In contrast to CBS, we define interleaving for broadcast and internal changes of processes. *Inter* deals with internal actions. The internal actions are not visible to an external observer. In each state, many internal actions can be performed simultaneously without any effect on each other. A network terminates when all protocols deployed at the different nodes terminate ($Term_1$ and $Term_2$). Rule Par_1 explains when in two separate sub-networks a communication occurs; when there is no conflict between senders and receivers, they can be occurred simultaneously. This rule resolves the hidden problem in MAC layer. The hidden problem occurs when two processes broadcast a message simultaneously while they are not connected to each other (they are hidden from each other). A collision is occurred at a node located at the intersection of radio transmission area of two nodes. So the collision management service provided by the *Media Access Control* (MAC) layer is considered implicitly by Par_1. Rule Par_2 explains a bunch of internal actions can be performed simultaneously in parallel with a communication in a network. Rule Par_3 allows defining locality for an action performed in a network. For instance when a communication occurs in a sub-network, it can be perform without forcing other nodes to be involved with. So other nodes can ignore the message if they are even be connected to the sender.

Table 3. Semantics of broadcast communication

$$\frac{P\downarrow}{[\![P]\!]_A\downarrow}:Term_1 \qquad \frac{P\xrightarrow{b}P'}{[\![P]\!]_A\xrightarrow{\tau}[\![P']\!]_A}:Inter \qquad \frac{P\xrightarrow{md!}P'}{[\![P]\!]_A\xrightarrow{\{md![A]\}}[\![P']\!]_A}:Bro_1$$

$$\frac{P\xrightarrow{mx?(md?)}P'}{[\![P]\!]_A\xrightarrow{md?\{A\}}[\![P'[d/x](P')]\!]_A}:Bro_2 \qquad \frac{N_1\xrightarrow{md?\sigma_1}N_1'\ \ N_2\xrightarrow{md?\sigma_2}N_2'}{N_1\,||\,N_2\xrightarrow{md?\sigma_1\cup\sigma_2}N_1'\,||\,N_2'}:Bro_3$$

$$\sigma\in\Gamma(A),\frac{N_1\xrightarrow{\{md![A]\}}N_1'\ \ N_2\xrightarrow{md?\sigma}N_2'}{N_1\,||\,N_2\xrightarrow{\{md![A]\}}N_1'\,||\,N_2'}:Bro_4 \qquad \frac{N_1\downarrow\ N_2\downarrow}{N_1\,||\,N_2\downarrow}:Term_2$$

$$\forall md![A_i]\in\beta.\Gamma(A_i)\cap\Gamma(A)=\varnothing,\frac{N_1\xrightarrow{\beta}N_1'\ \ N_2\xrightarrow{\{md![A]\}}N_2'}{N_1\,||\,N_2\xrightarrow{\beta\cup\{md![A]\}}N_1'\,||\,N_2'}:Par_1$$

$$\frac{N_1\xrightarrow{\tau}N_1'\ \ N_2\xrightarrow{\beta}N_2'}{N_1\,||\,N_2\xrightarrow{\beta}N_1'\,||\,N_2'}:Par_2 \qquad \frac{N_1\xrightarrow{\alpha}N_1'}{N_1\,||\,N_2\xrightarrow{\alpha}N_1'\,||\,N_2}:Par_3$$

The broadcast semantics supports *input blocking/output non-blocking behavior*. Input blocking and output non-blocking behavior indicates that when a protocol broadcasts a message to its peer, it does not block or wait for reception or even delivery of the message (Bro_1). In contrast, when a protocol requires an input message, it is blocked until the reception and delivery of the required message (Bro_4).

2.3. Mobility in Semantics

In this section, we will explain how the resulted labeled transition system induced by a network, can model mobility changes implicitly. As explained before, a pre-condition is assigned to each transition. These pre-conditions are the key point in modeling mobility implicitly in semantics, as the resulted labeled transition system is a symbolic representation of the network behavior. In other words, it abstractly models behavior of the network; instead of defining what movements are possible and consequently what are the next possible underlying topologies and network behaviors, it imposes a restriction over Γ. To explain it more clearly, we walk through an example. Consider three protocols $req0!.P_1$, $reqx?.P_2$ and $req1!.P_3$ deployed at nodes A, B and C respectively, where $req \in Message$, $0,1 \in Data$ and $x \in Variables$. The first node $[\![req0!.P_1]\!]_A$ sends the message $req0$ and then behaves as P_1. At the initial state $[\![req0!.P_1]\!]_A \,||\, [\![reqx?.P_2]\!]_B \,||\, [\![req1!.P_3]\!]_C$, there are two senders and one receiver. Thus there are seven possible next states; 1) A broadcasts and no node synchronized with, 2) A broadcasts, and only B receives, 3) C broadcasts and no node synchronized with, 4) C broadcasts, and only B receives, *5*) A and C broadcast simultaneously while there is no node in their common range and no one synchronized with, 6) A and C broadcast simultaneously, but B synchronized with A, and finally 7) A and C broadcast simultaneously, but B synchronized with C. We have shown the deduction tree for behaviors 1,2,5 and 6 in Table 4 (similar trees can be derived for behaviors 3,4 and 7 by exchanging A and C transitions). It should be noted that we have illustrated the behavior 6 in two deduction trees; in the second one, the first deduction tree should be replaced by (I). The pre-conditions of behaviors 1,2,5 and 6 are $\varnothing$, $\{B\} \in \Gamma(A)$, $\Gamma(A) \cap \Gamma(C) = \varnothing$ and finally $\{B\} \in \Gamma(A) \wedge \Gamma(A) \cap \Gamma(C) = \varnothing$ respectively.

The pre-condition in each transition is representative of many possible movements available between nodes. Instead of containing rules in semantics to allow mobility between nodes which change the underlying topology, we have modeled it implicitly in semantics. For example for the second behavior in Table 4, the pre-condition $\{B\} \in \Gamma(A)$ is representative of possible transitions in which nodes can move arbitrary such that the connections between them change but however in all B is connected to A (which is 32 topologies; 4 possible connectivity between B and C, 2 between A and B and 4 between A and C) as shown in Figure 1.

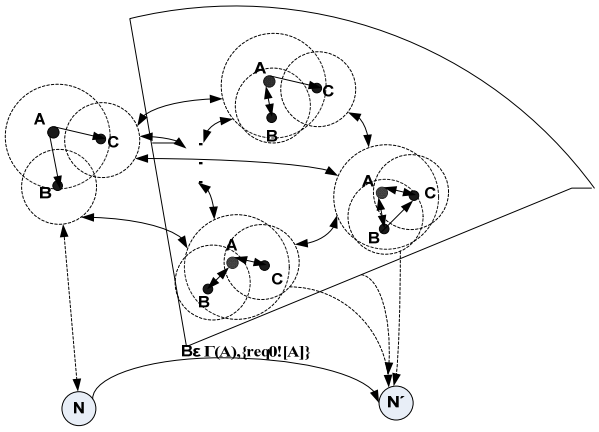

Figure 1. Exploration of a transition in induced labeled transition system of N

Table 4. The possible next states of network $[\![req0!.P_1]\!]_A \,||\, [\![reqx?.P_2]\!]_B \,||\, [\![req1!.P_3]\!]_C$

$$\frac{req0!P_1 \xrightarrow{req0!} P_1}{[\![req0!P_1]\!]_A \xrightarrow{\{req0![A]\}} [\![P_1]\!]_A} : Bro_1$$
$$\frac{}{N \xrightarrow{\{req0![A]\}} [\![P_1]\!]_A \,||\, [\![reqx?.P_2]\!]_B \,||\, [\![req1!.P_3]\!]_C} : Par_3$$

$$\{B\} \in \Gamma(A), \frac{\dfrac{req0!P_1 \xrightarrow{req0!} P_1}{[\![req0!P_1]\!]_A \xrightarrow{\{req0![A]\}} [\![P_1]\!]_A} : Bro_1 \quad \dfrac{reqx?P_2 \xrightarrow{req0?} P_2[0/x]}{[\![reqx?P_2]\!]_B \xrightarrow{req0?\{B\}} [\![P_2]\!]_B} : Bro_2}{N \xrightarrow{\{req0![A]\}} [\![P_1]\!]_A \,||\, [\![P_2[0/x]]\!]_B \,||\, [\![req1!.P_3]\!]_C} : Bro_4$$

$$\Gamma(A) \cap \Gamma(C) = \varnothing, \frac{\dfrac{req0!P_1 \xrightarrow{req0!} P_1}{[\![req0!P_1]\!]_A \xrightarrow{\{req0![A]\}} [\![P_1]\!]_A} : Bro_1 \quad \dfrac{req1!P_3 \xrightarrow{req1!} P_3}{[\![req1!P_3]\!]_C \xrightarrow{\{req1![C]\}} [\![P_3]\!]_C} : Bro_2}{N \xrightarrow{\{req0![A], req1![B]\}} [\![P_1]\!]_A \,||\, [\![reqx?P_2]\!]_B \,||\, [\![P_3]\!]_C} : Par_1$$

$$\{B\} \in \Gamma(A), \frac{\dfrac{req0!P_1 \xrightarrow{req0!} P_1}{[\![req0!P_1]\!]_A \xrightarrow{\{req0![A]\}} [\![P_1]\!]_A} : Bro_1 \quad \dfrac{reqx?P_2 \xrightarrow{req0?} P_2[0/x]}{[\![reqx?P_2]\!]_B \xrightarrow{req0?\{B\}} [\![P_2]\!]_B} : Bro_2}{[\![req0!P_1]\!]_A \,||\, [\![reqx?P_2]\!]_B \xrightarrow{\{req0![A]\}} [\![P_1]\!]_A \,||\, [\![P_2[0/x]]\!]_B} : Bro_4 : (I)$$

$$\Gamma(A) \cap \Gamma(C) = \varnothing, \frac{(I) \quad \dfrac{req1!P_3 \xrightarrow{req1!} P_3}{[\![req1!P_3]\!]_C \xrightarrow{\{req1![C]\}} [\![P_3]\!]_C} : Bro_1}{N \xrightarrow{\{req0![A], req1![B]\}} [\![P_1]\!]_A \,||\, [\![P_2[0/x]]\!]_B \,||\, [\![P_3]\!]_C} : Par_1 : (II)$$

As it can be seen in this figure, each transition is valid under a stronger pre-condition. This intuition has been formalized in the following proposition:

Proposition 1: Let $N \xrightarrow{\beta} N'$ and Pre_1 be the pre-condition assigned to this transition, then for all Pre_2 such that $\mathrm{Pre}_1 \Rightarrow \mathrm{Pre}_2$, $N \xrightarrow{\beta} N'$ holds.

It should be noted that a transition can be fired, if there is at least a topology which can satisfy its pre-condition. The pre-condition that there is at least one topology that satisfies it, is called a *valid pre-condition*.

3. Network Bisimulation

Informally, two closed protocols are behaviorally equivalent if for all valuations, both can perform the same actions and the resulting processes are again behaviorally equivalent. Two broadcast communicating processes (networks) are behaviorally equivalent if for all topologies (modeled by pre-conditions), both can perform the same actions, and the resulting processes are behaviorally equivalent.

A binary relation *Rp* over closed protocols is a protocol simulation if given $P\ R_p\ Q$ implies:

- if $P \xrightarrow{a} P'$ and a is an internal or a send action or receiving a data, then for some Q', $Q \xrightarrow{a} Q'$ and $P'\ R_p\ Q'$;
- if $P \xrightarrow{mx?} P'$ then, for each value $d \in Data$ for some Q', $Q \xrightarrow{mx?} Q'$and , $P'[d/x]\ \mathrm{R_p}\ Q'[d/x]$ holds;
- if $P\downarrow$, then $Q\downarrow$.

The relation R_p is a protocol bisimulation if both R_p and its inverse are protocol simulations. P is *protocol simulated* by Q, $P <_\mathrm{p} Q$, if there exists a protocol simulation R_p such that $P\ R_p\ Q$. P and Q are *protocol bisimilar*, $P \sim_\mathrm{p} Q$, if there exists a protocol bisimulation R_p such that $P\ R_p\ Q$. Protocol bisimilarity is similar to early bisimilarity defined for π-calculus [4], but since we have only considered closed terms, the congruency problems (name instantiation with matching [28]) does not occur here.

Let $\mathrm{N_1}$ and $\mathrm{N_2}$ be broadcast communicating processes. A Binary relation R_n over closed networks is a network simulation if given $N_1\ R_n\ N_2$, the following conditions hold:

- if $N_1 \xrightarrow{\beta} N_1'$ where $\beta \neq \tau$ and $\mathrm{Pre_1}$ is a valid pre-condition assigned to the transition, then for some $\mathrm{N_2}'$ and pre-condition $\mathrm{Pre_2}$, $\mathrm{N_2} \rightarrow^* \xrightarrow{\beta} \rightarrow^* N_2'$ such that $\mathrm{Pre_1} \Rightarrow \mathrm{Pre_2}$ and $\mathrm{N_1'\ R_n}\ N_2'$, where $\rightarrow^*$ is reflexive and transitive closure of $\xrightarrow{\tau}$ transitions;
- if $N_1 \xrightarrow{\tau} N_1'$, then for some N_2', $N_2 \rightarrow^* N_2'$ such that $N_1'\ \mathrm{R_n}\ \mathrm{N_2}'$;
- If $N_1\downarrow$, then $N_2\downarrow$.

The relation R_n is a network bisimulation if both R_n and its inverse are network simulations. N_1 is *network simulated* by N_2, $N_1 <_\mathrm{n} N_2$, if there exists a network simulation R_n such that $N_1\ R_n\ N_2$. N_1 and N_2 are *network bisimilar*, $N_1 \sim_\mathrm{n} N_2$, if there exists a network bisimulation R_n such that $N_1\ R_n\ N_2$. The definition of transition system has been given in *early* style and consequently the network bisimulation contemplates the substitution of the bound variables of an input with all possible values (rule Bro_2).

It can be shown that network bisimulation is an equivalence relation and is a congruence relation over network terms.

Theorem 1: Network bisimilarity is an equivalence relation.

Proof. The relation $R_n=\{(N,N)|N\in \mathrm{RPA}\}$ is obviously a network bisimulation relation. Assume that $N_1 \sim_\mathrm{n} N_2$ and R_n is network bisimulation relation witnessing it, then the relation $R_\mathrm{n}'=\{(N_2',N_1')|(N_1',N_2')\in R_n\}$ is a network bisimulation relation as well which implies $N_2 \sim_\mathrm{n} N_1$. Assume that $N_1 \sim_\mathrm{n} N_2$ and $N_2 \sim_\mathrm{n} N_3$ for states N_1,N_2,N_3. Let R_{n_1} and R_{n_2} be network bisimulation relations witnessing them respectively. Then we show that the relation composition $R_{n_1} \circ R_{n_2}$ is a network bisimulation implying $N_1 \sim_\mathrm{n} N_3$. Suppose $N_1 \xrightarrow{\beta} N_1'$, then there are two cases:

- If $\beta=\tau$, then as $N_1 \sim_\mathrm{n} N_2$, there is a N_2' such that $N_2 \rightarrow^* N_2'$ such that $(N_1',N_2') \in R_{n_1}$. We rewrite $N_2 \rightarrow^* N_2'$ as $N_2 \xrightarrow{\tau} N_{2_1}' \dots \xrightarrow{\tau} N_{2_m}' \xrightarrow{\tau} N_2'$ then as $N_2 \sim_\mathrm{n} N_3$, we have $N_3 \rightarrow^* N_{3_1}' \rightarrow^* \dots \rightarrow^* N_{3_m}' \rightarrow^* N_3'$ such that $\forall 2 \le i \le m.(\mathrm{N\_}\{N_{2_i}', N_{3_i}'\}) \in R_{n_2}$ and $(N_2',N_3') \in R_{n_2}$. So if $N_1 \xrightarrow{\tau} N_1'$ then $N_3 \rightarrow^* N_3'$ such that $(N_1',N_2') \in R_{n_1} \wedge (N_2',N_3') \in R_{n_2} \Rightarrow (N_1',N_3') \in R_{n_1} \circ R_{n_2}$.
- If $\beta \neq \tau$, so consider the pre-condition $\mathrm{Pre_1}$ has assigned to the transition. As $N_1 \sim_\mathrm{n} N_2$, then for some N_2' and pre-condition $\mathrm{Pre_2}$, $N_2 \rightarrow^* \xrightarrow{\beta} \rightarrow^* N_2'$ such that $\mathrm{Pre_1} \Rightarrow \mathrm{Pre_2}$. We rewrite $\rightarrow^*$ as sequences of τ transition. So similar to the above case, for some $\mathrm{Pre_3}$ and N_3' we have $N_3 \rightarrow^* \xrightarrow{\beta} \rightarrow^* N_3'$ such that $\mathrm{Pre_2} \Rightarrow \mathrm{Pre_3}$ and $(N_2',N_3') \in R_{n_2}$. Consequently if $N_1 \xrightarrow{\beta} N_1'$ then $N_3 \rightarrow^* \xrightarrow{\beta} \rightarrow^* N_3'$ such that $(\mathrm{Pre_1} \Rightarrow \mathrm{Pre_2} \wedge \mathrm{Pre_2} \Rightarrow \mathrm{Pre_3}) \Rightarrow (\mathrm{Pre_1} \Rightarrow \mathrm{Pre_3})$ and as $(N_1',N_2') \in R_{n_1} \wedge (N_2',N_3') \in R_{n_2} \Rightarrow (N_1',N_3') \in R_{n_1} \circ R_{n_2}$.

If $N_1\downarrow$ then $N_2\downarrow$ and consequently $N_3\downarrow$.

Theorem 2: Let $P_\mathrm{i} \sim_\mathrm{p} Q_\mathrm{j}$ and $N_\mathrm{i} \sim_\mathrm{n} M_\mathrm{i}$ for all $1 \le i,\mathrm{j} \le \mathrm{K}$ where P_i and Q_i are protocols and N_i and M_i are network processes, then:

- $a.P_\mathrm{i} \sim_\mathrm{p} a.Q_\mathrm{i}$
- $P_\mathrm{i}+P_\mathrm{j} \sim_\mathrm{p} Q_\mathrm{i}+\mathrm{Q_j}$
- $P_\mathrm{i} \bowtie_s P_\mathrm{j} \sim_\mathrm{p} \mathrm{Q\_i} \bowtie_s \mathrm{Q\_j}$, for arbitrary set of actions s
- $[B]P_\mathrm{i} \sim_\mathrm{p} [B]Q_\mathrm{i}$
- $P_\mathrm{i}/h \sim_\mathrm{p} Q_\mathrm{i}/h$, for arbitrary set of actions h
- $[\![P_i]\!]_{A_i} \sim_\mathrm{n} [\![Q_i]\!]_{A_i}$
- $N_1 \| \dots \| N_\mathrm{K} \sim_\mathrm{n} M_1 \| \dots \| M_\mathrm{K}$

Proof. The proofs for the protocol operators are straightforward. We only sketch the proof for broadcast communication. By assumption $N_\mathrm{i}\ R_{n_i}\ M_\mathrm{i}$ for $i=1,\dots,\mathrm{K}$. Let R_n denote $\{(N_1\|\dots\|N_\mathrm{K}, M_1\|\dots\|M_\mathrm{K}) \mid N_\mathrm{i}\ R_{n_i}\ M_\mathrm{i} \text{ for } 1 \le i \le \mathrm{K}\}$.

Suppose $N_1\|\dots\|N_\mathrm{K} \xrightarrow{\beta} N'$, we distinguish three cases:

1. If $\beta=\tau$, then using rules $Inter_2$ and Par_3, there are a set of networks namely I, each performs a τ transition. Without loss of generality suppose N_i is a network in this set, i.e. $N_\mathrm{i} \xrightarrow{\tau} N_\mathrm{i}'$. As $N_\mathrm{i}\ R_{n_i}\ M_\mathrm{i}$, then there is M_i' such that $M_\mathrm{i} \rightarrow^* M_\mathrm{i}'$ and $N_\mathrm{i}'\ R_{n_i}\ M_\mathrm{i}'$. Thus for each network in I like N_i, there is a network M_i such that by application of rule Par_3 we have $M_1\|\dots\|M_\mathrm{i}\|\dots\|M_\mathrm{K} \rightarrow^* M_1\|\dots\|M_\mathrm{i}'\|\dots\|M_\mathrm{K}$. By performing the sequences of τ transition in $M_1\|\dots\|M_\mathrm{K}$ for each τ transition involved in $N_1\|\dots\|N_\mathrm{K}$, results in $M_1\|\dots\|M_\mathrm{K} \rightarrow^* M'$ where $N'\ R_\mathrm{n}\ M'$.

2. If $\beta=md?\sigma$, then by using rules Bro_2, Bro_3 and Par_3, there are a set of nodes indicated by σ each performs a

receive transition. Thus for a subset of nodes like $\sigma_1 \subseteq \sigma$ in network j, we have $N_j \xrightarrow{md?\sigma_1} N_j'$. As $N_j R_{n_j} M_j$, there is M_j' such that $M_j \rightarrow^* \xrightarrow{md?\sigma_1} \rightarrow^* M_j'$ and $N_j' R_{n_j} M_j'$. Thus by application of rules Bro_2, Bro_3, Par_3 and Par_2, we have similarly $M_1||\ldots||M_K \rightarrow^* \xrightarrow{md?\sigma} \rightarrow^* M'$ and $N' R_n M'$.

3. If β is a list of send actions like $\{m_1 d_1![A_1],\ldots,m_n d_n![A_n]\}$ where each sender belongs to a network like N_j. The pre-condition assigned to this transition is Pre. Thus by application of rule Par_1, each network like N_j can perform an action like β_j where $\beta_j \subseteq \beta$ with pre-condition Pre_j such that $Pre_j \Rightarrow Pre$. The pre-condition Pre_j indicates for each A_i and A_j in β_j, $\Gamma(A_i) \cap \Gamma(A_j) = \varnothing$ holds. As $N_j R_{n_j} M_j$, there is M_j' and pre-condition Pre_j' such that $M_j \rightarrow^* \xrightarrow{md?\sigma_1} \rightarrow^* M_j'$ and $N_j' R_{n_j} M_j'$ and $Pre_j \Rightarrow Pre_j'$. By application of rules Par_2 and Par_1 $M_1||\ldots||M_K \rightarrow^* \xrightarrow{\beta} \rightarrow^* M'$and $N' R_n M'$ with pre-condition Pre' which is the conjunction of all Pre_j' and consequently $Pre = /\backslash Pre_j \Rightarrow /\backslash Pre_j' = Pre'$.

The last case to consider, is when $N_1||\ldots||N_K\downarrow$. By application of rule $Term_2$, this happen when for all $1 \le i \le K$, $N_i\downarrow$. Thus as $N_i R_{n_i} M_i$, then $M_i\downarrow$ holds and consequently $M_1||\ldots||M_K\downarrow$. Thus R_n satisfies the transfer conditions in network bisimilarity and thus R_n is a network bisimulation relation.

4. Cluster Based Routing Protocol

Cluster based routing protocol (CBRP) is a routing protocol designed for use in mobile ad hoc networks. This protocol divides the nodes of the ad hoc network into a number of overlapping or disjoint 2-hop-diameter clusters in a distributed manner. By clustering nodes into groups, the protocol efficiently minimizes the flooding traffic during route discovery and speeds up this process as well. In this Section, we use RPA to model cluster formation process of CBRP [29]. First we briefly explain the cluster formation process of CBRT and then model and examine its behavior using our algebra.

4.1. Cluster Formation Process of CBRT

The goal of cluster formation process is to impose some kind of structure or hierarchy in the otherwise completely disorganized ad hoc network. Each cluster has only one head and a number of member nodes and is uniquely identified by its *id*. Thus the status of a node in a cluster is either a cluster head or a member. A node not yet decided to join a cluster is in undecided status. A node can be a member of several clusters simultaneously. Nodes detect the presence of other nodes and organize themselves into clusters by using regular broadcast message known as "*hello*". Each node periodically broadcasts its status and of its neighbors information in *hello* messages. Each node has a neighbor table to maintain up to two-hop neighbors information using *link/connection status sensing mechanism* to update its information appropriately. We abstract away of this part in our model and only focus on communication behavior of process.

Initially all nodes are in undecided status, and all nodes commence their *hello* message broadcasts. A timer is also started at each node. If a node receives a broadcast from a head before the time-out, then it becomes a member. If it times out without hearing such a broadcast, then it goes from the undecided to head status. To assist nodes in moving from undecided to member, a head node will, in addition to its periodic broadcast, send out a triggered *hello* message whenever it receives a broadcast form an undecided node. Once all nodes are in either head or member states, cluster formation is complete. Cluster maintenance follows much the same as cluster formation. If a member node loses its connection to a head, then it will return to the undecided state and follows the initial procedure. If a head receives a *hello* message from another head, it will schedule a timer. After the timer times out, if the node was still connected to the head node, the node with smaller *id* remains as head and the other one becomes its member.

4.2. Specification of Cluster Formation Process

The cluster formation process, modeled by recursive variable CFP_{mid}, consists of four recursive variables (subprocess); three of them are called U_{mid}, H_{mid} and $M_{mid,id}$, specifying behavior of a node in undefined, head and member states respectively. The fourth recursive variable $A_{id,s}$ periodically broadcast the state and identifier of a node.

Table 5. The specification of cluster formation process in CBRT

$$CFP_{mid} = U_{mid} \bowtie_{s=\{stop\}} A_{mid,u}$$

$$U_{mid} = (hello\ id,s?.([s=h](stop.M_{mid,id}) + [s \neq h]U_{mid})) + stop.H_{mid}$$

$$H_{mid} = H_{mid}' \bowtie_{s=\{stop\}} A_{mid,h}$$

$$H_{mid}' = hello\ id,s?.([s=u]hello\ mid,h!.H_{mid}') + ([s=h]t_{sh}.t_{eh}.(hello\ id,s?.([mid<id]M_{mid,id}) + [mid>id]H_{mid}') + H_{mid}')$$

$$M_{mid,mh} = (hello\ mh,h?.M_{mid,mh}) + CFP_{mid}$$

$$A_{id,s} = hello\ id,s!.A_{id,s} + stop.0$$

When a node is in undefined state, modeled by U_{mid}, non-deterministically become a header (because its timer has expired), or receive a message from another header and then becomes its member (by calling $M_{mid,id}$). The H_{mid} composed of two other subprocesses namely H_{mid}' and $A_{mid,h}$ synchronized on action *stop*; H_{mid}' models responsive behavior of node in head state while $A_{mid,h}$ models the recursive behavior of node in head state, broadcasting its identifier and state. The recursive variable H_{mid}' non-deterministically behave as follows: it may receive a message from an undefined state node, or connect to a node which is header to. When a header, receives a message from another header, it runs a timer (modeled by action t_{sh}, after the timer times out (modeled by action t_{eh}), if the node was still connected, i.e.\ receives a message from that node again, by comparing its identifier by the other header (modeled by guarded command [*mid*<*id*]) decides its next state. When a node is in member states, it checks whether it is connected to its header. A member node may non-deterministically

disconnect from its header and then behave as an undefined node. It should be noted that we have modeled recursive variable's parameters by adding subscription to the variable, so for each concrete value A,B and C, there are recursive variables U_A, U_B and U_C. In other words, the specification in Table 5 is a *schematic* specification and for each concrete value, the corresponding process specifications exist.

4.3. Formal Analysis of Cluster Formation Process

In this Section, we are going to formally examine behavior of cluster formation process. Consider a network composed of three nodes, each deploying cluster formation process. The set of identifiers are chosen from *Names*, A,B and C ($A<B<C$). The two concrete values $u,h \in Data$ model state status values indicating undefined and header respectively. We are going to examine following properties for the network $[\![CFP_A]\!]_A \,||\, [\![CFP_B]\!]_B \,||\, [\![CFP_C]\!]_C$:

- The network behaves correctly, i.e. always one header is defined. For ease we use $[\![a.P_i]\!]_A$ to denote a process $P_i=a.P_i$ deployed at a node $[\![P_i]\!]_A$. The correct behaviors of the network includes the followings:

1. $N_1 \equiv [\![hello\ A, h!.P_1]\!]_A \,||\, [\![hello\ B, h!.P_2]\!]_B \,||\, [\![hello\ C, h!.P_3]\!]_C$
2. $N_2 \equiv [\![hello\ A, h!.P_1]\!]_A \,||\, [\![hello\ A, h?.P_2]\!]_B \,||\, [\![hello\ C, h!.P_3]\!]_C$
3. $N_3 \equiv [\![hello\ A, h!.P_1]\!]_A \,||\, [\![hello\ C, h?.P_2]\!]_B \,||\, [\![hello\ C, h!.P_3]\!]_C$
4. $N_4 \equiv [\![hello\ B, h!.P_1]\!]_A \,||\, [\![hello\ B, h?.P_2]\!]_B \,||\, [\![hello\ C, h!.P_3]\!]_C$
5. $N_5 \equiv [\![hello\ C, h?.P_1]\!]_A \,||\, [\![hello\ B, h!.P_2]\!]_B \,||\, [\![hello\ C, h!.P_3]\!]_C$
6. $N_6 \equiv [\![hello\ A, h!.P_1]\!]_A \,||\, [\![hello\ B, h!.P_2]\!]_B \,||\, [\![hello\ A, h?.P_3]\!]_C$
7. $N_7 \equiv [\![hello\ A, h!.P_1]\!]_A \,||\, [\![hello\ B, h!.P_2]\!]_B \,||\, [\![hello\ B, h?.P_3]\!]_C$
8. $N_8 \equiv [\![hello\ A, h!.P_1]\!]_A \,||\, [\![hello\ A, h?.P_2]\!]_B \,||\, [\![hello\ A, h?.P_3]\!]_C$
9. $N_9 \equiv [\![hello\ B, h?.P_1]\!]_A \,||\, [\![hello\ B, h!.P_2]\!]_B \,||\, [\![hello\ B, h?.P_3]\!]_C$
10. $N_{10} \equiv [\![hello\ C, h?.P_1]\!]_A \,||\, [\![hello\ C, h?.P_2]\!]_B \,||\, [\![hello\ C, h!.P_3]\!]_C$

For instance the second behavior denotes a network in which nodes A and C are headers and node B is a member of A and the last one denotes a network in which node C is a header and nodes A and B are its member. Thus to validate the behavior of the network, it suffices to examine

$$\forall 1 \le i \le 10. N_i <_n [\![CFP_A]\!]_A \,||\, [\![CFP_B]\!]_B \,||\, [\![CFP_C]\!]_C$$

- It is not possible to have a network without any header. Such behavior is defined by following networks:

1. $M_1 \equiv [\![hello\ B, h?.P_1]\!]_A \,||\, [\![hello\ C, h?.P_2]\!]_B \,||\, [\![hello\ B, h?.P_3]\!]_C$
2. $M_2 \equiv [\![hello\ B, h?.P_1]\!]_A \,||\, [\![hello\ A, h?.P_2]\!]_B \,||\, [\![hello\ B, h?.P_3]\!]_C$
3. $M_3 \equiv [\![hello\ C, h?.P_1]\!]_A \,||\, [\![hello\ C, h?.P_2]\!]_B \,||\, [\![hello\ B, h?.P_3]\!]_C$
4. $M_4 \equiv [\![hello\ C, h?.P_1]\!]_A \,||\, [\![hello\ C, h?.P_2]\!]_B \,||\, [\![hello\ A, h?.P_3]\!]_C$
5. $M_5 \equiv [\![hello\ C, h?.P_1]\!]_A \,||\, [\![hello\ A, h?.P_2]\!]_B \,||\, [\![hello\ A, h?.P_3]\!]_C$
6. $M_6 \equiv [\![hello\ B, h?.P_1]\!]_A \,||\, [\![hello\ A, h?.P_2]\!]_B \,||\, [\![hello\ A, h?.P_3]\!]_C$
7. $M_7 \equiv [\![hello\ B, h?.P_1]\!]_A \,||\, [\![hello\ C, h?.P_2]\!]_B \,||\, [\![hello\ A, h?.P_3]\!]_C$
8. $M_8 \equiv [\![hello\ C, h?.P_1]\!]_A \,||\, [\![hello\ A, h?.P_2]\!]_B \,||\, [\![hello\ B, h?.P_3]\!]_C$

Thus to verify the behavior of the network it suffice to examine

$$\forall 1 \le i \le 10. M_i \not<_n [\![CFP_A]\!]_A \,||\, [\![CFP_B]\!]_B \,||\, [\![CFP_C]\!]_C$$

5. Conclusion and Future Work

In this paper, we have introduced a new process algebra, called Routing Process Algebra (RPA), based on CSP multi-way synchronization for broadcast communication and broadcast operator of CBS (modified to the local broadcast). We modeled the underlying infrastructure using a connectivity function. We have also modeled the mobility behavior of nodes implicitly in the semantics of broadcast using predicates on connectivity functions called valid pre-conditions; each valid pre-condition defines a set of possible movements between nodes. We gave an operational semantics for RPA, and defined equality behavior between protocols and network. Finally we illustrated the application of our algebra for formal modeling and analysis of cluster formation process in cluster based routing protocols. By application of our algebra, we are able to verify the behavior of (routing) protocols against topology changes.

As explained in Section 1, this work based on [26]. The difference between this work with its earlier one in [25] is in modeling of mobility, computed by a common characteristic i.e.\ pre-condition, which is more abstract and consequently the semantics and equivalence relations are completely different. The work in [27] provides a setting to reason about networks by equations and is based on branching bisimilarity while this setting supports modeling parallel composition with true concurrency semantics based on observational congruency. This setting is applicable to verify networks with some restrictions over connectivity function.

We are going to provide a modal logic to verify a set of properties which can be verified by using network simulation and bisimulation. In latter to verify a property, all behaviors of a network should be considered, while in the former we can focus on specific behavior of a network satisfied in a possible execution of the network (execution on its corresponding labeled transition system). We also plan to provide a tool to automate formal analysis of networks. For instance we can examine the relations discussed in Section 4.3.

References

[1] C. A. R. Hoare, "Communicating sequential processes," *Communications of ACM*, Vol. 21, No. 8, pp. 666-677, 1978.

[2] R. Milner, *Communication and Concurrency*, Prentice-Hall, 1989.

[3] J. A. Bergstra, and J. W. Klop, "Algebra of communicating processes with abstraction," *Theoretical Computer Science*, Vol. 37, pp. 21-77, 1985.

[4] R. Milner, J. Parrow, and D. Walker, "Calculus of mobile processes," *Information and Computation*, Vol. 100, No. 1, pp. 1-77, 1992.

[5] R. De Renesse, and A. H. Aghvami, "Formal verification of ad-hoc routing protocols using SPIN model checker," *Proc, 12th IEEE Mediterranean Electrotechnical Conf.,* pp. 1177-1182, 2004.

[6] O. Wibling, J. Parrow, and A. Pears, "Automatized verification of ad hoc routing protocols," *Proc, 24th IFIP WG6.1 Int. Conf. on Formal Techniques for Networked and Distributed Systems*, Vol. 3235 of LNCS, pp. 343-358, Springer, 2004.

[7] K. Bhargavan, D. Obradovid, and C. A. Gunter, "Formal verification of standards for distance vector routing protocols," *Journal of the ACM*, Vol. 49, No. 4, pp. 538-576, 2002.

[8] P. Khengar, and A. H. Aghvami, "Warp - the wireless adaptive routing protocol," *Proc, IST Mobile Communications Summit*, pp. 480-485, 2001.

[9] C. Tschudin, R. Gold, O. Rensfelt, and O. Wibling, "LUNAR: a lightweight underlay network ad-hoc routing protocol and implementation," *Proc, Next Generation Teletraffic and Wired/Wireless Advanced Networking*, pp. 300, 2004.

[10] O. Wibling, J. Parrow, and A. Pears, "Ad hoc routing protocol verification through broadcast abstraction," *Proc, 25th IFIP WG6.1 Int. Conf. on Formal Techniques for Networked and Distributed Systems*, Vol. 3731 of LNCS, pp. 128-142, 2005.

[11] J. C. Godskesen, and O. Gryn, "Modeling and verification of security protocols for ad hoc networks using UPPAAL," *Proc, 18th Nordic Workshop on Programming Theory*, 2006.

[12] A. K. McIver, and A. Fehnker, "Formal techniques for analysis of wireless network," *Proc, 2nd IEEE-EASST Int. Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, pp. 263-270, IEEE, 2006.

[13] S. Chiyangwa, and M. Kwiatkowska, "A timing analysis of AODV," *Proc, 7th IFIP Int. Conf. on Formal Methods for Open Object-based Distributed Systems*, Vol. 3535 of LNCS, pp. 306-321, 2005.

[14] T. Razafindralambo, and F. Valois, "Performance evaluation of backoff algorithms in 802.11 ad-hoc networks," *Proc, Int. Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pp 82-89, ACM, 2006.

[15] K. V. S. Prasad, "A calculus of broadcasting systems," *Science of Computer Programming*, Vol. 25, No. 2-3, pp. 285-327, 1995.

[16] P. Sewell, P. Wojciechowski, and B. Pierce, "Location independence for mobile agents," *Proc, Workshop on Internet Programming Languages*, Vol. 1686 of LNCS, pp. 1-31, Springer, 1999.

[17] D. P. Gruska, and A. Maggiolo-Schettini, "Process algebras for network communication," *Fundamenta Informaticae*, Vol. 45, No. 4, pp. 359-378, 2001.

[18] G. Ciobanu, and K. N. Sridhar, "Specifications and verification of network protocols by process Algebra," *Proc, 7th Int. Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 250-258, 2005.

[19] P. A. Patsouris, "Algebraic modeling of an ad hoc network for mobile computing," *Journal of Parallel Distribution and Computing*, Vol. 61, No. 7, pp. 884-897, 2001.

[20] S. Nanz, and C. Hankin, "A framework for security analysis of mobile wireless networks," *Theoretical Computer Science*, Vol. 367, Vol. 1, pp. 203-227, 2006.

[21] N. Mezzetti, and D. Sangiorgi, "Towards a calculus for wireless systems," *Electronic Notes Theoretical Computer Science*, Vol. 158, pp. 331-353, 2006.

[22] J. Chr. Godskesen, "A calculus for mobile ad hoc networks," *Proc, 9th Int. Conf. on Coordination Models and Languages*, Vol. 4467 of LNCS, pp. 132-150, Springer, 2007.

[23] M. Merro, "An observational theory for mobile ad hoc networks," *Electronic Notes Theoretical Computer Science*, Vol. 173, pp. 275-293, 2007.

[24] S. A. Smolka, A. Singh, and C. R. Ramakrishnan, "A process calculus for mobile ad hoc networks," *Proc, 10th Int. Conf. on Coordination Models and Languages*, Vol. 75, No. 6, pp. 440-469, 2008.

[25] F. Ghassemi, W. J. Fokkink, and A. Movaghar, "Restricted broadcast process theory," *Proc, 6th Conference on Software Engineering and Formal Methods*, pp. 345-354. IEEE, 2008.

[26] F. Ghassemi, and A. Movaghar, "Modeling routing protocols in adhoc networks," *Advances in Computer Science and Engineering*, Vol. 6, pp. 419-426, Springer, 2008.

[27] F. Ghassemi, W. J. Fokkink, and A. Movaghar, "Equational reasoning on ad hoc networks," *3rd Conf. on Fundamentals of Software Engineering*, Vol. 5163 of LNCS, pp. 113-128, 2009.

[28] D. Sangiorgi, "A theory of bisimulation for the pi-calculus," *Proc, the 4th Int. Conf. on Concurrency Theory*, pp. 127-142, Springer-Verlag, 1993.

[29] J. Li, M. Jiang, and Y. C. Tay, "Cluster based routing protocol, INTERNET-DRAFT," draft-ietf-manet-cbrp-spec-01.txt, July 1999.

Fatemeh Ghasssemi received her B.Sc. and M.Sc in Computer Engineering from Tehran and Isfahan University, Iran, in 2004 and 2005 respectively. She is currently PhD student at Sharif University of Technology. Her research interests are specification and verification, process algebra and equational reasoning, and performance evaluation.

Email: fghassemi@mehr.sharif.edu

Ali Movaghar is currently a professor at the department of Computer Engineering in Sharif University of Technology in Tehran, Iran where he joined first as an assistant professor in 1993. He received his B.S. in Electrical Engineering from the University of Tehran in 1977 and both M.S. and Ph.D. in Computer, Information and Control Engineering from the University of Michigan at Ann Arbor in 1979 and 1985, respectively. He visited INRIA in France in 1984, worked at AT&T Bell Laboratories during 1985-1986, and taught at the University of Michigan during 1987-1989. His main areas of interest are performance and dependability modeling, verification and validation, computer networks, and distributed real-time systems.
Email: movaghar@sharif.edu

Paper Handling Data:

Submitted: 23.01.2009
Received in revised form: 23.08.2009
Accepted: 14.10.2009
Corresponding author: Fatemeh Ghasssemi,
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.