COMPOSITIONAL ANALYSIS OF MOBILE NETWORK PROTOCOLS*

Peng Wu[1] and Dongmei Zhang[2,1]

[1]Lab of Computer Science, Institute of Software, Chinese Academy of Sciences
Beijing 100080, China
Email: wp@ios.ac.cn
[2]School of Computer Science & Technology, Beijing University of Posts and Telecommunications
Beijing 100876, China
Email: zhangdm@bupt.edu.cn

Abstract

The paper presents a compositional framework for modelling network protocols with symbolic transition graphs. The main advantages of the framework are that it can address dynamic network topologies without requiring additional facilities; and it can work out system models that preserve deadlock freedom, namely the deadlock freedom of a system model depends only on the deadlock freedom of its each task component. A case study with Mobile IP illustrates the effectiveness of the modelling framework. Moreover, with a model checking experiment, it can be shown that the framework can extend the capability of the model checker to deal with more complicated system models than can be dealt with by direct model checking.

1 Introduction

The proliferation of portable devices has led to a wide spread of mobile computing. Mobile IP [1] is a typical mobility management protocol that aims to maintain the seamless connectivity to the Internet for mobile devices.

Research efforts have been devoted to model and verify Mobile IP with formal methods, which can be generally categorized into two groups: one provides explicit notations for mobility, such as π-calculus [2, 3], Mobile UNITY [4], while the other is to apply state-based approaches with explicit transitions of movement, such as ASTRAL [5]. However previous work paid mainly attention to the routing mechanism of Mobile IP, while ignored its feature of mobility detection in the sense that a mobile device can move actively in a nondeterministic way without having to locate itself, while as specified in Mobile IP, the device should determine its location dynamically based on the network it is currently attached to.

This paper proposes to apply Symbolic Transition Graphs with Assignment(STGA)[6] to analyze the inherent mobility of Mobile IP without explicit mobility notations or explicit movement transitions. The main contributions of the paper are as follows.

Firstly, the paper presents a compositional framework for modelling network protocols with STGA, which features that

1. The framework does not directly identify global states of a protocol entity, but decomposes the entity as a set of communicating sequential tasks with a set of state variables. The model of the entity can be synthesized from the one of its each task in a parallel way. Furthermore, a system model constructed in the context of the framework can be proved to be deadlock-free if each of its tasks is deadlock-free.

2. The framework does not target only at the static network topologies. By modelling a complete network topology with all possible communication links, a dynamic network topology can be regarded as a run-time instance of the complete network topology. In this way, the framework can also address the challenge from dynamic network topologies without additional mobility facilities.

3. The framework supports explicit communication

[*] The work is supported by grants 60223005 and 60242002 from the National Natural Science Foundation of China, a grant from the Chinese Academy of Sciences and a grant from the EYTT of MOE.

with message passing strategy, which can naturally express the working mechanism of a network protocol.

Secondly, Mobile IP is taken as a case study to illustrate the effectiveness of the framework. A much more systematic model $MIP4$ of Mobile IP is presented based on the modelling paradigm of the framework.

$MIP4$ addresses the full scenario of mobile communication based on Mobile IP, including mobility detection, registration and routing. However, the direct model checking of the model is infeasible for its high state-space complexity. While by model checking each of its tasks, the deadlock freedom of $MIP4$ can be concluded naturally with the aforementioned characteristic of the framework. Therefore, the framework can extend well the capability of the model checker to deal with more complicated system models than can be dealt with by direct model checking.

The rest of paper is organized as follows. Section 2 proposes the general framework for modelling network protocols after briefly introducing STGA. Section 3 presents the model $MIP4$. Section 4 analyzes the deadlock freedom of $MIP4$ by model checking each of its task components. The paper is concluded in Section 5 with future work.

2 Modelling Network Protocols with STGA

This section will briefly introduce STGA and then propose a general compositional framework for modelling network protocols by using STGA.

2.1 STGA

The following syntactic categories will be used in the sequel:

- Val is a set of values ranged over by v;
- Var is a set of variables ranged over by x, y, z;
- Exp is a set of expressions over $Val \cup Var$, ranged over by e;
- $BExp$ is a set of boolean expressions ranged over by b;
- $Chan$ is a set of channel names ranged over by c;
- $Sub \subseteq (Var \times Exp)^*$ is a set of substitutions ranged over by σ. A substitution $\sigma \equiv [\bar{e}/\bar{x}]$ specifies the type-respecting mapping from the vector of n distinct variables $\bar{x}$ to the vector of n expressions $\bar{e}$. We will often take the liberty to refer to a substitution $[\bar{e}/\bar{x}]$ as an assignment $\bar{x} := \bar{e}$.
- Act is a set of actions ranged over by α, which can be a silent action τ, an input action $c?\bar{x}$ or an output action $c!\bar{e}$. The sets of free and bound variables of actions are defined as usual: $fv(c!\bar{e}) = fv(\bar{e})$, $bv(c?\bar{x}) = \{\bar{x}\}$ and $fv(\alpha) = bv(\alpha) = \emptyset$ in all the other cases. The set of channel names used in actions is defined by $chan(c!\bar{e}) = chan(c?\bar{x}) = \{c\}$, $chan(\tau) = \emptyset$. An action $c?\bar{x}$ is referred to as a complement to $c!\bar{e}$, and vice versa.

Definition 2.1.1 (STGA)

1. *A symbolic transition graph with assignment(STGA) is a rooted directed graph*

$$\mathcal{G} = (N, \Sigma, r)$$

where

   (a) *N is a set of nodes ranged over by n, m. Each node n is labelled with an finite set of free variables $fv(n) \subseteq Var$;*

   (b) *$\Sigma \subseteq N \times (BExp \times Sub \times Act) \times N$ is a set of edges, each labelled with a guarded action with assignments $(b, \bar{x} := \bar{e}, \alpha)$.*

   (c) *$r \in N$ is the root of the graph $\mathcal{G}$.*

2. *A STGA $\mathcal{G}$ is well formed if for any $(n, (b, \bar{x} := \bar{e}, \alpha), m) \in \Sigma$, $fv(b) \cup fv(\bar{e}) \subseteq fv(n)$, $fv(\alpha) \subseteq \{\bar{x}\}$ and $fv(m) \subseteq \{\bar{x}\} \cup bv(\alpha)$*

   We will write $n \xrightarrow{b,\bar{x}:=\bar{e},\alpha} m$ for such edge, which means m can be reached from n by performing α, whenever b holds at n, after the free variables $\bar{x}$ are assigned with values of $\bar{e}$ evaluated at n. In the sequel, all STGAs $\mathcal{G} = (N, \Sigma, r)$ are assumed to be well-formed and abbreviated as $\mathcal{G}(N)$ without causing any confusion.

3. *Let $s \in (BExp \times Sub \times Act)^*$, A is a set of actions,*

   (a) *$chan(A) = \bigcup_{\alpha \in A} chan(\alpha)$.*

   (b) *$chan(s) = \bigcup_{(b,\theta,\alpha)\in\{s\}} chan(\alpha)$.*

   (c) *$n \xrightarrow{b,\theta,\alpha}$ if there exists m such that $n \xrightarrow{b,\theta,\alpha} m$;*

   (d) *(b, θ, α) is admissible at n if $n \xrightarrow{b,\theta,\alpha}$;*

   (e) *$n \xrightarrow{s} n$ or $n \rightarrow n$ if s is empty;*

   (f) *$n \xrightarrow{s} m$ if $s = (b_0, \theta_0, \alpha_0) \ldots (b_k, \theta_k, \alpha_k)$, $k \geq 0$ and there exist $n_0, \ldots n_k \in N$ such that $n = n_0 \xrightarrow{b_0,\theta_0,\alpha_0} \cdots \xrightarrow{b_k,\theta_k,\alpha_k} n_k = m$;*

   (g) *$n \xrightarrow{s}$ if there exists $m \in N$ such that $n \xrightarrow{s} m$;*

(h) s is a path of $\mathcal{G}(N)$ if there exists $n \in N$ such that $n \xrightarrow{s}$.

(i) (b, θ, α) is eventually admissible from n, denoted by $n \overset{b,\theta,\alpha}{\Rightarrow}$, if there exists s and n' such that $n \xrightarrow{s} n' \xrightarrow{b,\theta,\alpha}$ and $chan(s) \cap chan(\alpha) = \emptyset$;

2.2 Modelling Network Protocols

A network protocol(NP) is a set of rules for computing entities to communicate with each other. It can be formally described as a tuple

$$NP = (\mathbb{E}, \mathbb{C}^e, \mathbb{C}^p)$$

where

1. $\mathbb{E}$ is a finite set of protocol entity arrays $E[1..r]$, each of which contains r protocol entities $E_i(\bar{x})$ indexed by i $(1 \leq i \leq r)$. Without causing any confusion, the set of protocol entities declared in a protocol entity array $E[1..r]$ is also denoted by E.

2. $\mathbb{C}^e$ is a finite set of channel arrays $c^e[1..n_e]$, each of which contains n_e channels c^e_j $(1 \leq j \leq n_e)$ that connect protocol entities with the external environment of NP. Such channel can be regarded as the abstraction of a service access point open for upper layer protocols or applications to call service primitives provided by NP, or for lower layer protocols to make service primitive callbacks.

3. $\mathbb{C}^p$ is a finite set of channels arrays $c^p[1..n_p]$, each of which contains n_p channels c^p_k $(1 \leq k \leq n_p)$ that connect protocol entities with each other. Such channel can be regarded as the abstraction of a communication link for protocol entities to exchange protocol data units.

Let $\mathbb{PE}$ denote all protocol entities defined in $\mathbb{E}$, that is,

$$\mathbb{PE} = \{E_i(\bar{x}) \mid \exists E[1..r] \in \mathbb{E}, 1 \leq i \leq r\}.$$

Each protocol entity in $\mathbb{PE}$ has to respond various incoming messages timely and continuously according to its local status, indicated by its state variables. Those messages may contain service primitive calls from upper layer protocols or applications, or protocol data units submitted by lower layer protocols. Therefore, two types of tasks are allocated for a protocol entity: one is to process incoming messages, while the other to control read/write access to (local) state variables. These tasks progress concurrently and cooperate with each other to fulfil the functionalities of the network protocol. Two types of factors should be identified for task division: incoming messages and state variables. A protocol entity $E_i(\bar{x})$ can be formally described as a tuple

$$E_i(\bar{x}) = (\mathbb{T}_i, \mathbb{O}_i, \mathbb{I}_i, \mathbb{V}_i, \mathbb{C}^l_i)$$

where

1. $\mathbb{T}_i$ is a finite set of task components $T_i(\bar{x})$, represented in STGA.

2. $\mathbb{O}_i$ is a finite set of channels, through which $E_i(\bar{x})$ sends messages to other protocol entities or to the environment.

3. $\mathbb{I}_i$ is a finite set of channels, through which $E_i(\bar{x})$ receives messages from other protocol entities or from the environment; For each $c \in \mathbb{I}_i$, there is a message task component $T_{i_c}(\bar{x}_c) \in \mathbb{T}_i$, such that

$$T_{i_c}(\bar{x}_c) = c?(\bar{m}).MR_{i_c}(\bar{x}'_c)$$

where $\{\bar{x}'_c\} \subseteq \{\bar{x}_c\} \cup \{\bar{m}\}$. Informally, $T_{i_c}(\bar{x}_c)$ receives a message $\bar{m}$ and then evolves as a message routine $MR_{i_c}(\bar{x}'_c)$, which responds the message according to the present status of $E_i(\bar{x})$ and then continues as $T_{i_c}(\bar{x}''_c)$.

4. $\mathbb{V}_i$ is a finite set of state variables. For each variable $v \in \mathbb{V}_i$, there is an access-control task component $T_{i_v}(\bar{x}_v)$ such that

$$T_{i_v}(\bar{x}_v) = c_r?(\bar{m}_r).R_{i_v}(\bar{x}'_v) + c_w?(\bar{m}_w).W_{i_v}(\bar{x}'_v)$$

where $c_r, c_w \in \mathbb{C}^l_i$. $R_{i_v}(\bar{x}'_v)$ responses to a read request on variable v (from other component in $E_i(\bar{x})$) with the value of v and then evolves as $T_{i_v}(\bar{x}_v)$; $W_{i_v}(\bar{x}'_v)$ responses to a write request on variable v (from other component in $E_i(\bar{x})$) by setting the value of v accordingly and then evolves as $T_{i_v}(\bar{x}''_v)$.

5. $\mathbb{C}^l_i$ is a finite set of local channels that connect components in $E_i(\bar{x})$ with each other. $\mathbb{C}^l_i \cap (\mathbb{O}_i \cup \mathbb{I}_i) = \emptyset$. Because each message task component progresses independently with other message task components, the local channels are only available for communications between message task components and access-control task components.

6. Any input action $c?(\bar{x})(c \in \mathbb{I}_i \cup \mathbb{C}^l_i)$ is not guarded because of the following aspects:

   - a protocol entity should be reactive to external stimuli at any time;

- a read or write quest on a state variable should be admitted at any time;
- any response for a previous read request should be dealt with by the sender of the request.

7. Let $\mathbb{C}$ denote all channels defined in $\mathbb{C}^e$ and $\mathbb{C}^p$, that is,

$$\begin{aligned}\mathbb{C} \;=\; & \{c^e_j \mid \exists c^e[1..n_e] \in \mathbb{C}^e, 1 \leq j \leq n_e\} \\ & \cup\{c^p_k \mid \exists c^p[1..n_p] \in \mathbb{C}^p, 1 \leq k \leq n_p\},\end{aligned}$$

Then $\mathbb{C} = \bigcup_{E_i(\bar{x}) \in \mathbb{PE}} (\mathbb{O}_i \cup \mathbb{I}_i)$

In this way, the protocol entity $E_i(\bar{x})$ can be defined as

$$E_i(\bar{x}) = (\prod_{c \in \mathbb{I}_i} T_{i_c}(\bar{x}_c) | \prod_{v \in \mathbb{V}_i} T_{i_v}(\bar{x}_v)) \backslash \mathbb{C}^l_i$$

and the network protocol NP can be defined as

$$NP = (\prod_{E_i(\bar{x}) \in \mathbb{PE}} E_i(\bar{x})) \backslash \mathbb{C}^p$$

2.3 Deadlock Freedom

Definition 2.3.1 (Persistent Reset Capability) *A STGA $\mathcal{G} = (N, \Sigma, r)$ is persistently resettable if for any node $n \in N$, there exists a path s such that*

$$n \xrightarrow{s} r$$

s is referred to as a resetting path of node n.

Definition 2.3.2 (Interoperability) *Let C be a set of channels, k STGAs $\mathcal{G}_1(N_1), \ldots, \mathcal{G}_k(N_k)$ are interoperable on C if for each $(n_1, \ldots, n_k) \in N_1 \times \cdots \times N_k$, $Act = \{\alpha \mid n_i \xrightarrow{b,\theta,\alpha}, 1 \leq i \leq k\}$,*

1. *$chan(Act) - C \neq \emptyset$, or*
2. *$chan(Act) \subseteq C$ and*
   (a) *there exists at least a $c \in C$, n_i and $n_j(1 \leq i,\ j \leq k)$ such that $n_i \xrightarrow{b,\theta_1,c!\bar{e}}$ and $n_j \overset{true,\theta_2,c?\bar{x}}{\Rightarrow}$. In this case, n_j is referred to as a communicating peer of n_i and (n_i, n_j) is referred to as a communicating pair;*
   (b) *among all communicating pairs in $(n_1, \ldots, n_k)$, there is no such sequence $(n_{i_0}, n_{i_1})(n_{i_1}, n_{i_2}) \cdots (n_{i_{k-1}}, n_{i_k})$ that $k > 0$ and $i_k = i_0$.*

Compared against [7], the definition 2.3.2 is much more simple and feasible in that no semantic checking is required for the existence of a communicating pair. The basic ideas behind the definition 2.3.2 rest in

1. $b \models true$ holds always;
2. A communicating pair is defined based on an output action bcasue its symmetric form can be ignored due to the following fact that if $chan(Act) \subseteq C$, then there exists a $c \in C$, n_i and $n_j(1 \leq i,\ j \leq k)$ such that $n_i \xrightarrow{b,\theta_1,c!\bar{e}}$ and $n_j \overset{true,\theta_2,c?\bar{x}}{\Rightarrow}$, if and only if there exists a $c \in C$, n_i and $n_j(1 \leq i,\ j \leq k)$ such that $n_i \xrightarrow{true,\theta_1,c?\bar{x}}$ and $n_j \overset{b,\theta_2,c!\bar{e}}{\Rightarrow}$. Note that $true \models b$ does not hold always.

Lemma 2.3.3 *A protocol entity $E_i(\bar{x}) = (\mathbb{T}_i, \mathbb{O}_i, \mathbb{I}_i, \mathbb{C}^l_i, \mathbb{V}_i)$ is persistently resettable if and only if*

1. *for any $c \in \mathbb{I}_i$, $T_{i_c}(\bar{x}_c)$ is persistently resettable;*
2. *for any $v \in \mathbb{V}_i$, $T_{i_v}(\bar{x}_v)$ is persistently resettable;*
3. *for each $c \in \mathbb{I}_i$ $T_{i_c}(\bar{x}_c)$ and all $T_{i_v}(\bar{x}_v)$ are interoperable on $\mathbb{C}^l_i$.*

Proof

1. *For "if": Each T_{i_c} can evolve back to its root infinitely often. Each T_{i_v} can, too. Furthermore, whenever an action on a local channel in $\mathbb{C}^l_i$ is admissible, there will always be a communicating peer that can make the whole entity evolve forward by a communication on the local channel. So $E_i(\bar{x})$ can also evolve back to itself infinitely often, i.e. $E_i(\bar{x})$ is persistently resettable.*
2. *For "only if": By the definition above, each node n in the STGA of each task component $T(\bar{x})$ will be covered in the STGA of the protocol entity $E_i(\bar{x})$. Therefore the resetting path of node n can be constructed by projecting a resetting path of node $(\ldots, n, \ldots)$ in the STGA of $E_i(\bar{x})$ on T. In addition, the interoperability among task components of $E_i(\bar{x})$ can be safely reflected from the compositional generation of the STGA of $E_i(\bar{x})$.*

□

Lemma 2.3.4 *A network protocol $NP = (\mathbb{E}, \mathbb{C}^e, \mathbb{C}^p)$ is persistently resettable if and only if*

1. *for any $E_i(\bar{x}) \in \mathbb{PE}$, $E_i(\bar{x})$ is persistently resettable.*
2. *All $E_i(\bar{x})$ are interoperable on $\mathbb{C}^p$.*

Proof *Similarly to Lemma 2.3.3.* □

With Lemma 2.3.3 and 2.3.4, it can be proved that a persistently resettable network protocol preserves deadlock freedom.

Theorem 2.3.5 (Protocol Deadlock Freedom) *A persistently resettable network protocol is deadlock-free if each of its task components is deadlock-free.*

Proof *By Lemma 2.3.3 and 2.3.4.* □

3 Modelling Mobile IP

This section will illustrate a STGA model of Mobile IP in the context of the modelling framework proposed in Section 2. Formally, Mobile IP can be described as a tuple

$$MIP4^{n_{ha},n_{fa},n_{mh}} = (\mathbb{E}_4, \mathbb{C}_4^e, \mathbb{C}_4^p)$$

with

$$\mathbb{E}_4 = \{HA4[1..n_{ha}], FA[1..n_{fa}], MH4[1..n_{mh}]\}$$

where

1. $HA4$, FA and $MH4$ are the model for home agents, foreign agents and mobile hosts in Mobile IP, respectively;

2. n_{ha}, n_{fa}, and n_{mh} are the number of home agents, foreign agents and mobile hosts, respectively;

Table 2 describes how these entities are connected with the environment(denoted by -) via channels in $\mathbb{C}_4^e$, while Table 3 describes how these entities are connected with each other via channels in $\mathbb{C}_4^p$. Herein all channels in $\mathbb{C}_4^e \cup \mathbb{C}_4^p$ are unilateral. The entities that may send messages to a channel are listed in column From; while the ones that may receive messages from the channel is listed in column To. Due to the space of the paper, local channels for each entity are omitted.

1. Only a pair of channels *agt_sol* and *agt_adv* is defined so that a mobile host can send Agent Solicitation messages to and receive Agent Advertisement messages from any home or foreign agent. Such nondeterminism just reflects the liberty of host movement.

2. Suppose $N_{mh} = \{k \mid 1 \leq k \leq n_{mh}\}$, $N_{ha} = \{i \mid 1 \leq i \leq n_{ha}\}$, $hn : N_{mh} \rightarrow N_{ha}$ is a home network function that maps each mobile host MH_k to its home agent $HA_{hn(k)}$.

$MIP4$ has been checked to be persistently resettable so the deadlock freedom can be preserved.

3.1 Home Agent

A home agent contains five message task components: AA, RYF, RYM, ARP and RTA.

- AA may issue Agent Advertisement messages infinitely often and respond incoming Agent Solicitation messages accordingly. $HOME$ means it is a home agent.

- RYF and RYM deal with incoming Registration Request messages relayed from foreign agents and sent by mobile hosts, respectively.

- ARP may respond an incoming ARP Request message with the MAC address(mac) of the home agent if the requested host has moved outside.

- RTA may transfer an incoming datagram to its destination or to the corresponding foreign agent via a tunnel.

For a home agent, only one local variable is used, i.e. the binding list(*binding_list*) that saves the care-of addresses of mobile hosts. Task component HBC takes control of read/write access to *binding_list*.

Consequently, the home agent can be formally described as

$$\boxed{\begin{array}{ll} HA4_i(mac, ip, \mathit{binding\_list}) & = \\ & AA_i(ip, HOME) \mid RYF_i|RYM_i \\ & \mid \prod\limits_{hn(k)=i} ARP_k \mid RTA_i(mac, ip) \\ & |HBC_i(\mathit{binding\_list}) \end{array}}$$

3.2 Foreign Agent

A foreign agent contains four message task components: AA, RLM, RLH and TN.

- AA is just the same as the one for a home agent, except for its agent information $FOREIGN$, which means it is a foreign agent.

- RLM and RLH relay incoming Registration Request for mobile hosts or incoming Registration Reply messages for home agents, respectively.

- TN receives an incoming encapsulated datagram from a home agent via a tunnel and transfer the datagram to its destination accordingly.

For a foreign agent, only one local variable is used, i.e. the visitor list(*visitor_list*) that saves the visiting mobile hosts. Task component FBC takes control of read/write access to *visitor_list*.

Consequently, the foreign agent can be formally described as

$$\begin{aligned} FA_j(ip, visitor\_list) \quad = \quad & AA_j(ip, FOREIGN) \mid RLM_j(ip) \\ & |RLH_j(ip) \mid TN_j(ip) \\ & |FBC_j(visitor\_list) \end{aligned}$$

3.3 Mobile Host

A mobile host contains five message task components: AD, RRF, RRH, $MARP$ and $DT4$.

- AD determines the location of the mobile host according to the incoming Agent Advertisement messages. It may also issue Agent Solicitation messages actively. Once detecting the movement, it will inform the home agent of the mobile ho0st to update its binding status with a care-of address in a Registration Request message.
- RRF and RRH check incoming Registration Reply messages received from foreign or home agents, respectively and update its local status if the home agent accepts its registration or deregistration.
- $MARP$ responds an incoming ARP Request message, addressed to itself, with its own MAC address (mac) if it stays inside the home network.
- DT receives datagrams addressed to itself.

For a mobile host, two local variables are used, namely the movement flag and the pending request. The former is a boolean variable to indicate whether the mobile host has moved outside($true$) or not($false$). It is useful for $MARP$ to decide whether it should respond an incoming ARP Request message. The latter saves the last Registration Request messages sent by the mobile host. Registration Reply messages corresponding to Registration Request messages previous to the last one will all be ignored. LS and PR takes control of read/write access to these two variables, respectively. The movement flag is initialized as $false$ and the pending request none.

Note that the care-of address of the mobile host is only a parameter of AD, initialized as 0, because other message task components do not have to know the concrete location of the mobile host, but the status whether it has moved outside or not.

Consequently, the mobile host can be formally described as

$$\begin{aligned} MH4_k(ip, hip, mac) \quad = \quad & AD_{k,hn(k)}(ip, hip, 0) \mid RRF_k \mid RRH_k \\ & |MARP_k(mac) \mid DT4_{k,hn(k)}(ip) \\ & |LS_k(false) \mid PR_k(ip) \end{aligned}$$

Component	$\lvert S\rvert$	$\lvert N\rvert$	$\lvert E\rvert$	DF
AA	5	1	2	TRUE
RYF	395	3	5	TRUE
RYM	135	3	5	TRUE
ARP	40	4	6	TRUE
RTA	252	7	10	TRUE
HBC	434	7	14	TRUE
RLM	134	3	5	TRUE
RLH	278	5	9	TRUE
TN	179	6	9	TRUE
FBC	300	4	11	TRUE
AD	330	6	17	TRUE
RRF	265	4	7	TRUE
RRH	265	4	7	TRUE
DT	30	2	3	TRUE
MARP	20	4	5	TRUE
LS	28	3	6	TRUE
PR	1174	8	14	TRUE

Table 1: Results of Model Checking Each Component

4 Model Checking Mobile IP

A representative instance of $MIP4$ is considered in the case study, i.e. $MIP4^{1,1,1}$ that contains only one home agent. one foreign agent and one mobile host. Although it sounds simple, the model checking experiment shows that $MIP4^{1,1,1}$ is far more complicated than can be dealt with by the model checker[8]. However, the model checker can verify the deadlock freedom of its each task component. Therefore, the deadlock freedom of $MIP4^{1,1,1}$ can be concluded with Theorem 2.3.5.

The deadlock freedom can be expressed in the μ-calculus formula DF, which is admissible to the model checker.

$$DF \triangleq \nu X. <-> \ true \ \wedge \ [-]X$$

Table 1 illustrates the results of model checking each task component of $MIP4^{1,1,1}$ against DF. $|S|$ is the number of states that have been traversed during model checking. $|N|$ and $|E|$ are the number of nodes and edges of each STGA model, respectively.

5 Conclusion

This paper presents a symbolic and compositional framework for modelling network protocols with STGA. It inherits the layered nature of network protocols. The system model in the context of the framework is constituted of a series of task components that communicate with each other via message passing strategy. The case study on Mobile IP illustrates the effectiveness of the modelling framework. The main advantage of the framework lies in

1. It can address the case of dynamic network topology without additional syntactic or pragmatic facilities.

2. It preserves the deadlock freedom so that the deadlock freedom of the system model depends only on the deadlock freedom of its each task component. In this way, the framework can extend the capability of the model checker to deal with more complicated system models than can be dealt with by direct model checking.

As future work, the framework can be extended for parameterized verification. The inherent parameterized modelling nature of the framework can help verify concrete network protocols in a more cost-effective way.

References

[1] Perkins, C.: IP mobility support. IETF RFC 2002 (1996)

[2] Amadio, R.M., Prasad, S.: Modelling IP mobility. In Sangiorgi, D., de Simone, R., eds.: Ninth International Conference on Concurrency Theory. Lecture Notes in Computer Science 1466, Nice, France, Springer-Verlag (1998) 301–316

[3] Wen, X., Hai, F., Hui-min, L.: Optimization and implementation of a bisimulation checking algorithm for the pi calculus. The Journal of Software **12** (2001) 159–166

[4] McCann, P.J., Roman, G.C.: Modeling mobile IP in mobile unity. ACM Transaction on Software Engineering and Methodology **8** (1999) 115–146

[5] Dang, Z., Kemmerer, R.A.: Using the ASTRAL model checker to analyze mobile IP. In: 21st international conference on Software engineering, Los Angeles, California, United States, IEEE Computer Society Press (1999) 132–141

[6] Lin, H.: Symbolic transition graph with assignment. In: CONCUR'96. Volume 1119 of Lecture Notes in Computer Science., Pisa, Italy (1996) 50–65

[7] Wu, P., Zhang, D.: Compositional analysis of mobile ip with symbolic transition graphs. In: 16th Internatioal Conference on Computer Communication. (2004) 1481–1488

[8] Lin, H.: Model checking value-passing processes. In: 8th Asia-Pacific Software Engineering Conference (APSEC'2001), Macau (2001)

Channel	**From**	**To**	**Description**
$send_i$	-	$HA4_i$	To receive the datagrams from the environment.$(1 \leq i \leq n_{ha})$
$location_k$	$MH4_k$	-	To indicate the location of the k-th mobile host.$(1 \leq k \leq n_{mh})$
$route_k$	$HA4_{hn(k)}$ FA	-	To indicate the routing path for a datagram addressed to k-th mobile host. $(1 \leq k \leq n_{mh})$

Table 2: Channels in $\mathbb{C}_4^e$

Channel	**From**	**To**	**Description**
agt_sol	$MH4$	$HA4$ FA	To send an agent solicitation
agt_adv	$HA4$ FA	$MH4$	To send an agent advertisement
$reg\_req\_hm_i$	$MH4$	$HA4_i$	To send a registration request directly to the i-th foreign agent.$(1 \leq i \leq n_{ha})$
$reg\_rep\_hm_k$	$HA4$	$MH4_k$	To send a registration reply directly to the k-th mobile host.$(1 \leq k \leq n_{mh})$
$reg\_req\_fm_j$	$MH4$	FA_j	To send a registration request via the j-th foreign agent.$(1 \leq j \leq n_{fa})$
$reg\_rep\_fm_k$	FA	$MH4_k$	To forward a registration reply to the k-th mobile host.$(1 \leq k \leq n_{mh})$
$reg\_req\_hf_i$	FA	$HA4_i$	To forward a registration request to the i-th home agent.$(1 \leq i \leq n_{ha})$
$reg\_rep\_hf_j$	$HA4$	FA_j	To send a registration reply via the j-th foreign agent.$(1 \leq j \leq n_{fa})$
$forward_k$	$HA4$ FA	$MH4_k$	To receive a datagram from a home or foreign agent.$(1 \leq k \leq n_{mh})$
$tunnel_j$	$HA4$	FA_j	To receive a tunnelled datagram from a home agent.$(1 \leq j \leq n_{fa})$

Table 3: Channels in $\mathbb{C}_4^p$