ISCAS-LCS-05-02 March, 2005

中国科学院软件研究所
计算机科学实验室报告

# Compositional Analysis of Mobile Network Protocols

by

**Peng Wu**
**Dongmei Zhang**

**Laboratory of Computer Science**
**Institute of Software**
**Chinese Academy of Sciences**
**Beijing 100080. China**

# Compositional Analysis of Mobile Network Protocols

Peng Wu [1,2]

*Lab of Computer Science, Institute of Software*
*Chinese Academy of Sciences*
*Beijing 100080, China*

Dongmei Zhang [3]

*School of Computer Science & Technology*
*Beijing University of Posts and Telecommunications*
*Beijing 100876, China*

**Abstract**

A compositional framework is proposed 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 mobility 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. Case studies with Mobile IPv4 and IPv6 illustrate the effectiveness of the modelling framework. The model checking experiments show 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. Moreover, some infrangibilities of Mobile IPv6 are disclosed in the sense that it can not maintain the binding coherency all the time, which may result in unreachable or unstable routes.

*Key words:* Compositional Analysis, Mobile IP, Model Checking

## 1 Introduction

The proliferation of portable devices has led to a wide spread of mobile computing. Mobility supports for IPv4 and IPv6, referred to as Mobile IPv4 and

[1] 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.
[2] Email: wp@ios.ac.cn
[3] Email: zhangdm@bupt.edu.cn

Mobile IPv6 respectively, were developed to maintain the seamless connectivity to the Internet for mobile devices [11,12,5,6].

Research efforts have been devoted to model and verify Mobile IPv4 with formal methods, which can be generally categorized into two groups: one provides explicit notations for mobility, such as $\pi$-calculus [1,14], Mobile UNITY [9], while the other is to apply state-based approaches with explicit transitions of movement, such as ASTRAL [2]. However previous work paid mainly attention to the routing mechanism of Mobile IPv4, 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 IPv4, the device should determine its location dynamically based on the network it is currently attached to.

Mobile IPv6 separates location discovery from the context of mobility management and simplifies the routing mechanism of Mobile IPv4. Research efforts have been devoted to analyze Mobile IPv6 quantitatively with respect to its performance. However, qualitative analysis of Mobile IPv6, with respect to its functionality, is of little concern.

This paper proposes to apply Symbolic Transition Graphs with Assignment(STGA) to analyze the inherent mobility of Mobile IP without explicit mobility notations or explicit movement transitions [16]. The main contributions of the paper are as follows.

Firstly, the paper presents a compositional framework for modelling network protocols with STGA [7], which features that

(i) 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.

(ii) 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.

(iii) The framework supports explicit communications by message passing strategy, which can naturally express the working mechanism of a network protocol, while [9,2] support only implicit communications via shared variables and [1,14] consider only name passing at a higher level of abstraction.

Secondly, Mobile IPv4 and IPv6 are taken as examples to illustrate the effectiveness of the framework. $MIP4$ and $MIP6$ are presented based on the modelling paradigm of the framework, which can address the full sce-

nario of mobile communication based on Mobile IPv4 and IPv6 respectively, including mobility detection(for Mobile IPv4 only), registration, routing and ARP-related issues.

These two case studies are conducted in a model checker for value-passing concurrent system recently developed in the Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences [8,18]. The tool accepts a system description in value-passing CCS and a property expressed as a first-order $\mu$-calculus formula, and tries to verify that the system satisfies the property. In case the system fails to enjoy the required property, an informative diagnosis message will be generated in the form of an execution sequence, explaining the cause of the failure.

Unfortunately, it is infeasible to directly verify $MIP4$ and $MIP6$ by model checking due to their high state space complexities. While by model checking each of their tasks, the deadlock freedom of $MIP4$ and $MIP6$ 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. Moreover, the model checking experiments also disclose the infrangibility of Mobile IPv6 in the sense that it can not maintain the binding coherency all the time, which may result in unreachable or unstable routes.

**Related works**

[4] presents a comprehensive summary on protocol design, validation and verification techniques that regard network protocols as finite state machines or other variants. Recently [3] conducts a systematic case study on the i-protocol with explicit-state model checking techniques, while [10] illustrates how to effectively find errors in large network protocol implementations using model checking techniques with reasonable effort. The main purpose of our study is to address inherent dynamic characteristics of mobile network protocols using a hierarchical state-oriented approach and identify a deadlock-free design pattern to alleviate state space explosion problem of 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 case studies with Mobile IPv4 and IPv6. The paper is concluded in Section 4 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 with 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 $\sigma$. 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 $\alpha$, which can be a silent action $\tau$, 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.

### *2.1 STGA*

**Definition 2.1** [STGA]

(i) A symbolic transition graph with assignment(STGA) is a rooted directed graph $\mathcal{G} = (N, \Sigma, r)$, where
  (a) $N$ is a finite set of nodes ranged over by $n, m$. Each node $n$ is associated 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 finite 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 $\mathcal{G}$.

(ii) 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 $\alpha$, 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.

(iii) 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) $(b, \theta, \alpha)$ is admissible at $n$ if $n \xrightarrow{b,\theta,\alpha}$;
  (e) $n \xrightarrow{s} n$ or $n \to n$ if $s$ is empty;
  (f) $n \xrightarrow{s} m$ if $s = (b_0, \theta_0, \alpha_0) \dots (b_k, \theta_k, \alpha_k)$, $k \geq 0$ and there exist $n_0, \dots n_k \in N$ 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) $(b, \theta, \alpha)$ is eventually admissible from $n$, denoted by $n \xrightarrow{b,\theta,\alpha}$, if there exists $s$ and $n'$ such that $n \xrightarrow{s} n' \xrightarrow{b,\theta,\alpha}$ and $chan(s) \cap chan(\alpha) = \emptyset$;

Let $\prod_i P_i$ denote a parallel composition of a finite number of processes $P_i (i \geq 0)$. The semantics of value-passing CCS and STGA are out of the scope of the paper. For more detailed information one can refer to [7].

### *2.2 Modelling Network Protocols*

A network protocol(NP) is a set of rules for computing entities to communicate with each other. These protocol entities are categorized into a finite number of groups, where each member shares the same functionality definition. Thus, the network protocol can be formally described as a tuple

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

where

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

(ii) $\mathbb{C}^e$ 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.

(iii) $\mathbb{C}^p$ is a finite set of channel 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 to 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 protocol entities. Therefore, two types of tasks are allocated for a protocol entity: one is to process incoming messages, while the other to control read/write accesses 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. Thus, a protocol entity $E_i(\bar{x})$ can be formally described as a tuple

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

where

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

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

(iii) $\mathbb{I}_i$ 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 to the message according to the present status of $E_i(\bar{x})$ and then continues as $T_{i_c}(\bar{x}''_c)$.

(iv) $\mathbb{C}_i^l$ is a finite set of local channels that connect components in $E_i(\bar{x})$ with each other. $\mathbb{C}_i^l \cap (\mathbb{O}_i \cup \mathbb{I}_i) = \emptyset$. 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.

(v) $\mathbb{V}_i$ 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 $c_r, c_w \in \mathbb{C}_i^l$. $R_{i_v}(\bar{x}'_v)$ 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 $T_{i_v}(\bar{x}''_v)$.

(vi) Any input action $c?(\bar{x})(c \in \mathbb{I}_i \cup \mathbb{C}_i^l)$ is not guarded because of the following aspects:
- A protocol entity should be reactive to external stimuli at any time;
- A read or write request on a state variable should be admitted at any time;
- Any response to a previous read request should be dealt with by the sender of the request.

(vii) Let $\mathbb{C}$ denote all channels defined in $\mathbb{C}^e$ and $\mathbb{C}^p$, that is,

$\mathbb{C} = \{c_j^e \mid \exists c^e[1..n_e] \in \mathbb{C}^e, 1 \leq j \leq n_e\} \cup \{c_k^p \mid \exists c^p[1..n_p] \in \mathbb{C}^p, 1 \leq k \leq n_p\}$

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 a composition of those message task components and access-control components with local

channels among them restricted from being accessed externally, that is,

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

Similarly, the network protocol $NP$ can be defined as a composition of its each protocol entity with peer-to-peer channels among them restricted from being accessed externally, that is,

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

With [7] it can be easily seen that models of these resulted protocol entities and the network protocol are also represented in STGA. Thus, by top-down task division and bottom-up composition, a network protocol can be formally described in STGA with a solid semantic foundation for further analysis and verification. Note that the way to combine those components or entities is through parallel composition(plus hiding) because otherwise a sequential structure will result in deadlock of the resulted model [15].

### *2.3 Deadlock Freedom*

Interactions among various components of a network protocol engender higher difficulty in analyzing the network protocol from a systematic viewpoint than in analyzing each of its components separately. As far as deadlock freedom is concerned, deadlock-free entities cannot naturally result in a deadlock-free network protocol, neither can deadlock-free components result in a deadlock-free protocol entity, due to potential mismatches in peer-to-peer interactions among protocol entities or components. In this section, a formal concept of interoperability is proposed to represent a smooth cooperation among various participants in a network protocol or a protocol entity.

**Definition 2.2** [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 $(n_1,\ldots,n_k)\in N_1\times\cdots\times N_k$, $Act=\{\alpha \mid n_i \xrightarrow{b,\theta,\alpha}, 1\le i\le k\}$,

(i) $chan(Act)-C\neq\emptyset$, or

(ii) $chan(Act)\subseteq C$ and

(a) there exists at least a $c\in C$, $n_i$ and $n_j(1\le i,\ j\le k)$ such that $n_i \xrightarrow{b,\theta_1,c!\bar{e}}$ and $n_j \xrightarrow{true,\theta_2,c?\bar{x}}$. 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 [16], the definition 2.2 is much more concise and feasible in that no semantic checking is required for the existence of a communicating pair. The basic ideas behind the definition 2.2 rest in

(i) $b \models true$ holds always;

(ii) A communicating pair is defined based on an output action because 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* $n_j \xRightarrow{true,\theta_2,c?\bar{x}}$, *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* $n_j \xRightarrow{b,\theta_2,c!\bar{e}}$. Note that $true \models b$ does not hold always.

**Theorem 2.3 (Deadlock Freedom of A Protocol Entity)** *A protocol entity is deadlock-free if*

(i) *Each of its task components is deadlock-free.*

(ii) *For each* $c \in \mathbb{I}_i$, $T_{i_c}(\bar{x}_c)$ *and all* $T_{i_v}(\bar{x}_v)$ *are interoperable on* $\mathbb{C}_i^l$.

**Proof.** By condition 1, each $T_{i_c}$ can evolve forever and each $T_{i_v}$ can, too. Then with condition 2, the protocol entity can evolve in one of the following two ways:

(i) The whole entity steps forward by performing an action on certain channel $c \notin \mathbb{C}_i^l$ if the action is admissible at one of its components, respecting the first case of Definition 2.2.

(ii) Whenever only an action on a local channel $c \in \mathbb{C}_i^l$ is admissible, there will always be a communicating peer that can make the whole entity evolve forward by a communication on the local channel, due to the second case of Definition 2.2.

So $E_i(\bar{x})$ can also evolve forever, i.e. $E_i(\bar{x})$ is deadlock-free. □

**Theorem 2.4 (Deadlock Freedom of A Network Protocol)** *A network protocol is deadlock-free if*

(i) *Each of its protocol entities is deadlock-free.*

(ii) *All* $E_i(\bar{x})$ *are interoperable on* $\mathbb{C}^p$.

**Proof.** Similarly to Theorem 2.3. □

## 3 Case Study

This section will present two cast studies with Mobile IPv4 and IPv6. To address non-trivial data structures involved in network protocols, STGA has been extended with arrays, lists and records [18]. In the sequel, the following syntactic notations will be used in STGA scripts:

(i) $c(\bar{x})$ and $\bar{c}\langle\bar{e}\rangle$ denote an input action $c?\bar{x}$ and an output action $c!\bar{e}$, respectively;

(ii) $v_a[i]$ denotes the $(i+1)$-th element of an array variable $v_a (i \geq 0)$.

(iii) $v_r \# v_m$ denotes the member variable $v_m$ in a record variable $v_r$.

For the detailed syntax of the STGA script language, one can refer to [18].

To concentrate on mobility supports for IPv4 and IPv6, those features irrelated to mobility management are not considered in our case studies. All model checking experiments are carried out on a SUN Enterprise 4500 Server with four 450MHz UltraSPARC processors and 2GB RAM.

### *3.1 Mobile IPv4*

Among various IP mobility proposals, Mobile IPv4 [11,12] is the oldest and probably the most widely known mobility management proposal with IP. It excels in its simplicity and scalability. Three types of entities are involved in Mobile IPv4: Mobile Host, Home Agent and Foreign Agent.

Mobile hosts are entities that are allowed to migrate around the IP network. Each mobile host is assumed to have a home network, from which it obtains a constant IP address, called its Home Address.

Home and foreign agents are introduced for mobility management. Each time a mobile host connects to a network at a new location, it will obtain a temporary address, called Care-of Address (COA) from a foreign agent in the local network. Then the mobile host must inform its home agent of the new address by a registration procedure, which begins when the mobile host, possibly with the assistance of the foreign agent, sends a registration request with the COA. When the home agent receives this request, it may typically add the necessary information to its routing table, approve the request, and send a registration reply back to the mobile host. Two types of COAs are defined, one is called Foreign Agent Care-of Address, that is, the IP address of the foreign agent; while the other is called Collocated Care-of Address, which is allocated by other address configuration mechanism.

Once the home agent is aware of the current address of the mobile host in roaming, it will tunnel towards the mobile host all packets destined for its home address, possibly with the assistance of the foreign agent. Indeed, these packets, normally routed, will obviously arrive at the home network where the home agent will intercept and encapsulate them towards the foreign agent. Then the foreign agent will decapsulate the traffic from the home agent and forward it to the mobile host.

Both the home agent and the foreign agent typically broadcast agent advertisements at regular intervals to make themselves known, the former for mobile hosts to discover it has returned to its home network, while the latter for mobile hosts to retrieve a COA. If a mobile host needs to get a COA and does not wish to wait for the periodic advertisement, it can broadcast or multicast a solicitation, which will then be answered by any foreign or home agent having received the solicitation.

*3.1.1 Modelling*

This section will illustrate a STGA model of Mobile IPv4 in the context of the modelling framework proposed in Section 2. Formally, Mobile IPv4 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

$$\mathbb{E}_4 = \{HA4[0..n_{ha}-1], FA[0..n_{fa}-1], MH4[0..n_{mh}-1]\}$$

where

(i) $HA4$, $FA$ and $MH4$ are models of home agents, foreign agents and mobile hosts in Mobile IP, respectively;

(ii) $n_{ha}$, $n_{fa}$, and $n_{mh}$ are the number of home agents, foreign agents and mobile hosts, respectively;

Let $i, j, k$ range over indexes of home agents, foreign agents and mobile hosts, respectively. Table 1 describes how these entities are connected with the environment(denoted by -) via channels in $\mathbb{C}_4^e$(row 1 to 4) and how these entities are connected with each other via channels in $\mathbb{C}_4^p$(row 5 to 14). Suppose $N_{mh} = \{k \mid 0 \leq k \leq n_{mh} - 1\}$, $N_{ha} = \{i \mid 0 \leq i \leq n_{ha} - 1\}$, $hn : N_{mh} \rightarrow N_{ha}$ is a home network function that maps each mobile host $MH4_k$ to its home agent $HA4_{hn(k)}$. 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. Local channels in each entity are just as their names imply.

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 agents. Such nondeterminism just reflects the liberty of host movement.

$MIP4$ can be easily proved to be interoperable so the deadlock freedom can be preserved.

**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 to incoming Agent Solicitation messages accordingly.

$$AA_i(ip, agt) = \overline{agt\_adv}\langle ip, agt, i\rangle.AA_i(ip, agt) + agt\_sol.AA_i(ip, agt) + \tau.AA_i(ip, agt)$$

  The action $\tau$ is used to simulate the delay of responses. The parameter $ip$ is the IP address of a home or foreign agent and $agt$ the type of the agent. For a home agent, $agt$ is initialized as $HOME$.

- $RYF$ and $RYM$ deal with incoming Registration Request messages relayed

| **Channel** | **From** | **To** | **Description** |
|---|---|---|---|
| $send_i$ | - | $HA4_i$ | To receive the datagrams from the environment. |
| $location_k$ | $MH4_k$ | - | To indicate the location of the $(k+1)$-th mobile host. |
| $route_k$ | $HA4_{hn(k)}, FA$ | - | To indicate the routing path for a datagram addressed to $(k+1)$-th mobile host. |
| $stderr$ | $FA, MH4$ | - | To indicate an exception. |
| $agt\_sol$ | $MH4$ | $HA4, FA$ | 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 $(i+1)$-th foreign agent. |
| $reg\_rep\_hm_k$ | $HA4$ | $MH4_k$ | To send a registration reply directly to the $(k+1)$-th mobile host. |
| $reg\_req\_fm_j$ | $MH4$ | $FA_j$ | To send a registration request via the $(j+1)$-th foreign agent. |
| $reg\_rep\_fm_k$ | $FA$ | $MH4_k$ | To forward a registration reply to the $(k+1)$-th mobile host. |
| $reg\_req\_hf_i$ | $FA$ | $HA4_i$ | To forward a registration request to the $(i+1)$-th home agent. |
| $reg\_rep\_hf_j$ | $HA4$ | $FA_j$ | To send a registration reply via the $(j+1)$-th foreign agent. |
| $forward_k$ | $HA4, FA$ | $MH4_k$ | To receive a datagram from a home or foreign agent. |
| $tunnel_j$ | $HA4$ | $FA_j$ | To receive a tunnelled datagram from a home agent. |

Table 1
Channels in $\mathbb{C}_4^e \cup \mathbb{C}_4^p$

from foreign agents and sent by mobile hosts, respectively.

$$RYF_i = reg\_req\_hf_i(ha, coa, j, k).$$
$$(\overline{reg\_rep\_hf_j}\langle ha, coa, OK\rangle.\overline{set\_hbinding_i}\langle k, ha, coa, j\rangle.RYF_i$$
$$+\overline{reg\_rep\_hf_j}\langle ha, coa, NOK\rangle.RYF_i)$$
$$RYM_i = reg\_req\_hm_i(ha, coa, k).$$
$$(\overline{reg\_rep\_hm_k}\langle ha, coa, OK\rangle.\overline{set\_hbinding_i}\langle k, ha, coa, -1\rangle.RYM_i$$
$$+\overline{reg\_rep\_hm_k}\langle ha, coa, NOK\rangle.RYM_i)$$

The parameter *ha* is the home address of a mobile host and *coa* the COA that *ha* is associated with. $OK$ indicates the acceptance of a request, while $NOK$ the rejection of it.

Requests from foreign agents are to register COAs, while those from mobile hosts are to deregister them. If a request is accepted, the current COA of the mobile host will be updated(in $RYF$) or cancelled(in $RYM$) via the channel *set_hbinding*.

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

$$ARP_k(mac) = arp\_req\_agt_k(ip).\overline{hld\_req_i}\langle ip\rangle.$$
$$(hld\_rep\_out_i(coa, j).\overline{arp\_rep_k}\langle mac\rangle.ARP_k(mac)$$
$$+hld\_rep\_in_i(port).ARP_k(mac) + hip\_unknown_i.ARP_k(mac))$$

The channel *hld_req* is used to query the location of a mobile host, which may be out of its home network, indicated by the channel *hld_rep_out*, or in its home network, indicated by the channel *hld_rep_in*. The channel *hld_unknown* indicates the host is unknown to the home agent. The parameter *port* is the index of the link between the home agent and the mobile host in the home network.

Herein the MAC address of the home agent is assumed to be known. ARP requests on the home agent itself are discarded, which would not affect the analysis result. Note that *ARP* supports only for proxy ARP, while gratuitous ARP is not considered.

- *RTA* may transfer an incoming datagram to its destination or to the corresponding foreign agent via a tunnel(*tu_ha*). The parameter *imac* is the target MAC address, *src* the source IP address, *tgt* the target IP address and *payload* the data portion of the datagram. Herein, it is assumed that

(i) A home agent is also a gateway in the home network. *pkt_out* is just a routing channel.

(ii) Only datagrams destined for mobile hosts, which the home agent serves, are considered. Others towards the home agent itself or other mobile hosts are all discarded, which would not affect the analysis result.

$$
\begin{aligned}
RTA_i(mac, ip) = {} & send_i(imac, src, tgt, payload). \\
& if\ (imac = mac)\ then \\
& \quad \overline{hld\_req_i}\langle tgt\rangle. \\
& \quad (hld\_rep\_out_i(coa, j). \\
& \quad \overline{tu\_ha_j}\langle ip, coa, src, tgt, payload\rangle.ack_i.RTA_i(mac, ip) \\
& \quad + hld\_rep\_in_i(k).\overline{\mathbf{route_k}}\langle \mathbf{ip}\rangle. \\
& \quad \overline{pkt\_out_k}\langle src, tgt, payload\rangle.ack_i.RTA_i(mac, ip) \\
& \quad + hip\_unknown_i.RTA_i(mac, ip)) \\
& else\ RTA_i(mac, ip)
\end{aligned}
$$

For a home agent, only one local variable is defined, that is, the binding list(*binding_list*) that saves the care-of addresses of mobile hosts. Each binding record consists of a home address($m\_ha$), the COA that the home address is associated with($m\_coa$) and the index of the link between the home agent and corresponding foreign agent($m\_fidx$). Task component $HBC$ takes control of read/write accesses to *binding_list*.

$$
\begin{aligned}
& HBC_i(cache) = \\
& set\_hbinding_i(idx, ha, coa, j). \\
& cache[idx]\#m\_ha := ha. \\
& cache[idx]\#m\_coa := coa. \\
& cache[idx]\#m\_fidx := j. \\
& BC_i(cache)) \\
& + \\
& hld\_req_i(ip). \\
& CheckHBC_i(cache, 0, ip)
\end{aligned}
\qquad
\begin{aligned}
& CheckHBC_i(cache, iidx, ip) = \\
& if\ (iidx = CACHESIZE)\ then \\
& \quad \overline{hip\_unknown_i}.HBC_i(cache) \\
& else\ ( \\
& \quad if\ (cache[iidx]\#m\_ha = ip)\ then( \\
& \quad\quad if\ (cache[iidx]\#m\_coa = 0)\ then \\
& \quad\quad\quad \overline{hld\_rep\_in_i}\langle cache[iidx]\#m\_idx\rangle. \\
& \quad\quad\quad HBC_i(cache) \\
& \quad\quad else \\
& \quad\quad\quad \overline{hld\_rep\_out_i}\langle cache[iidx]\#m\_coa, \\
& \quad\quad\quad\quad cache[iidx]\#m\_fidx\rangle. \\
& \quad\quad\quad HBC_i(cache)) \\
& \quad else \\
& \quad\quad CheckHBC_i(cache, iidx + 1, ip))
\end{aligned}
$$

Consequently, the home agent can be formally described as

$$HA4_i(mac, ip, binding\_list) = AA_i(ip, HOME) \mid RYF_i | RYM_i \mid \prod_{hn(k)=i} ARP_k \mid RTA_i(mac, ip) \mid HBC_i(binding\_list)$$

**Foreign Agent**

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

- $AA$ is the same as the one for a home agent, except for a foreign agent, $agt$ is initialized as $FOREIGN$. Herein the foreign agent COA is just the IP address of the foreign agent.
- $RLM$ deals with incoming Registration Request messages from mobile hosts. On receiving a Registration Request message, it may reject the registration directly by sending a Registration Reply message with $NOK$ as a status flag, or relay the registration request to given home agent, of which the index is $i$, and then set the local status of the registration as $ISPENDING$ via the channel $set\_pending$.

$$\begin{aligned} RLM_j(ip) = {} & reg\_req\_fm_j(ha, coa, i, k). \\ & if\ (coa = ip)\ then\ ( \\ & \quad \overline{reg\_rep\_fm_k}\langle ha, coa, NOK\rangle.RLM_j(ip) \\ & \quad + \overline{reg\_req\_hf_i}\langle ha, coa, j, k\rangle.\overline{set\_pending_j}\langle k, ha\rangle.RLM_j(ip)) \\ & else\ RLM_j(ip) \end{aligned}$$

  The condition $coa = ip$ requires the request to be received from the mobile host that the foreign agent serves.

- $RLH$ deals with incoming Registration Reply messages from home agents.

$$\begin{aligned} RLH_j(ip) = {} & reg\_rep\_hf_j(ha, coa, rep). \\ & if\ (coa = ip)\ then\ ( \\ & \quad \overline{fld\_req_j}(ha, ISPENDING) \\ & \quad (fip\_unknown_j.RLH_j(ip) + fld\_rep\_in_j.RLH_j(ip) \\ & \quad + fld\_rep\_out_j(k). \\ & \quad if\ (rep = OK)\ then \\ & \quad\quad \overline{set\_fbinding_j}\langle k, ha\rangle.\overline{reg\_rep\_fm_k}\langle ha, coa, rep\rangle.RLH_j(ip) \\ & \quad else\ \overline{reg\_rep\_fm_k}\langle ha, coa, rep\rangle.RLH_j(ip)) \\ & else\ RLH_j(ip) \end{aligned}$$

  On receiving a Registration Reply message, where the COA is the IP address of the foreign agent, it will query the local status of the registration via the channel $fld\_req$. If the registration is pending, indicated by

$fld\_req\_out$ with $k$, the index of the mobile host that sent the registration, then the Registration Reply message will be relayed to that mobile host and if the status flag $rep$ in the reply is OK, the local status of the registration is updated as $ISREADY$ via channel $set\_fbinding$.

- $TN$ receives an incoming encapsulated datagram from a home agent via a tunnel and transfers the datagram to its target mobile host, which has registered the foreign agent successfully with the home agent.

$$TN_j(ip) = tu\_ha_j(tsrc, ttgt, src, tgt, payload).\overline{fld\_req_j}\langle tgt, ISREADY\rangle.$$
$$(fld\_rep\_out_j(k).\overline{\mathbf{route_k}}\langle \mathbf{ip}\rangle.\overline{pkt\_out_k}\langle src, tgt, payload\rangle.TN_j$$
$$+fld\_rep\_in_j.\overline{stderr}\langle 3\rangle.TN_j + fip\_unknown_j.\overline{stderr}\langle 3\rangle.TN_j)$$

The parameter $tsrc$ and $ttgt$ are the source and the target IP address of the encapsulated datagram respectively, while $src$ and $tgt$ are the source and the target IP address of the source datagram, respectively. The error No. 3, reported via the channel $std\_err$, indicates that the endpoint of the tunnel has not been registered by the target mobile host.

For a foreign agent, only one local variable is defined, that is, the visitor list($visitor\_list$) that saves the visiting mobile hosts. The home address($m\_ha$) and the registration status $m\_tag$ of each visiting mobile host can be referred to via its index. Task component $FBC$ takes control of read/write accesses to $visitor\_list$.

$FBC_j(cache) =$
$set\_pending_j(idx, ha).$
$cache[idx]\#m\_ha := ha.$
$cache[idx]\#m\_tag := ISPENDING.$
$FBC_j(cache))$
$+set\_fbinding_j(idx, ha).$
$cache[idx]\#m\_ha := ha.$
$cache[idx]\#m\_tag := ISREADY.$
$FBC_j(cache))$
$+$
$fld\_req_i(ip, tag).$
$CheckFBC_j(cache, 0, ip, tag)$

$CheckFBC_j(cache, iidx, ip, tag) =$
$if\ (iidx = CACHESIZE)\ then$
  $\overline{fip\_unknown_j}.FBC_j(cache)$
$else\ ($
  $if\ (cache[iidx]\#m\_ha = ip)\ then($
    $if\ (cache[iidx]\#m\_tag = tag)\ then$
      $\overline{fld\_rep\_out_j}\langle iidx\rangle.$
      $FBC_j(cache))$
    $else$
      $\overline{fld\_rep\_in_j}.FBC_j(cache)$
  $else$
    $CheckFBC_j(cache, iidx + 1, ip, tag))$

Consequently, the foreign agent can be formally described as

$$FA_j(ip, visitor\_list) = AA_j(ip, FOREIGN) \mid RLM_j(ip) \mid RLH_j(ip)$$
$$\mid TN_j(ip) \mid FBC_j(visitor\_list)$$

**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 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 host to update its binding status with a care-of address in a Registration Request message.

$$
\begin{aligned}
AD_{k,i}(ip, hip, coa) = agt\_adv(icoa, agt, j).( \\
& if\ (coa = icoa)\ then \\
& \quad AD_{k,i}(ip, coa) \\
& else\ ( \\
& \quad if\ (hip = icoa\ and\ agt = HOME)\ then \\
& \quad\quad \overline{pending\_req_k}\langle 0, false\rangle. \\
& \quad\quad \overline{reg\_req\_hm_i}\langle ip, 0, k\rangle.AD_k(ip, hip, coa) \\
& \quad else \\
& \quad\quad \overline{pending\_req_k}\langle icoa, true\rangle. \\
& \quad\quad \overline{reg\_req\_fm_j}\langle ip, icoa, i, k\rangle.AD_{k,i}(ip, hip, coa)))) \\
& + set\_careof_k(ncoa). \\
& \quad \overline{\mathbf{location_k}}\langle \mathbf{ncoa}\rangle.AD_{k,i}(ip, hip, ncoa) \\
& + \overline{agt\_sol}.AD_{k,i}(ip, hip, coa)
\end{aligned}
$$

  The parameter $hip$ is the IP address of its home agent, $coa$ the current COA, $icoa$ the COA advertised by a home or foreign agent. $coa = icoa$ means the mobile host does not change its attaching point to a local network. An Agent Advertisement message received from its home agent indicates the mobile host has moved back to its home network, which requires its current COA to be cancelled. Other Agent Advertisement messages indicate the mobile host has moved out of the local network, which requires its current COA to be updated. Once the registration request is approved, indicated via the channel $set\_careof$ with a new COA($ncoa$), the mobile host will set $ncoa$ as its current COA.

- $RRF$ and $RRH$ check incoming Registration Reply messages received from the home agent and from foreign agents, respectively and update its local status if the home agent accepts its registration or deregistration. Because the mobile host has to determine its location via the latest Agent Advertisement message, only the reply corresponding to the last registration request is concerned.

$$
\begin{array}{l|l}
RRF_k = & RRH_k = \\
reg\_rep\_fm_k(ha, coa, rep). & reg\_rep\_hm_k(ha, coa, rep). \\
if\ (rep = OK)\ then & if\ (rep = OK)\ then \\
\quad \overline{is\_last_k}\langle ha, coa\rangle. & \quad \overline{is\_last_k}\langle ha, coa\rangle. \\
\quad (is\_last\_yes_k.\overline{set\_careof_k}\langle coa\rangle.RRF_k & \quad (is\_last\_yes_k.\overline{set\_careof_k}\langle coa\rangle.RRH_k \\
\quad + is\_last\_no_k.RRF_k & \quad + is\_last\_no_k.RRH_k \\
else\ RRF_k & else\ RRH_k
\end{array}
$$

On receiving a positive reply($rep = OK$), $RRF$ and $RRH$ will query for the last request via the channel $is\_last$. The channel $is\_last\_yes$ indicates the reply is just the one for the last request, which therefore make the current COA updated with the one involved in the reply($coa$) via the channel $set\_careof$; while the channel $is\_last\_no$ indicates the reply is for an outdated request. All negative relies will be ignored because no change has to be made for the mobile host.

- $MARP$ responds to an incoming ARP Request message, addressed to itself, with its own MAC address ($mac$) if it is in the home network.

$$
\begin{aligned}
MARP_k(mac) = arp\_req\_host_k.\overline{ls\_req_k}. \\
(ls\_rep\_home_k.\overline{arp\_rep_k}\langle mac\rangle.MARP_k(mac) \\
+ls\_rep\_out_k.MARP_k(mac))
\end{aligned}
$$

The channel $ls\_req$ is used to query the status of the mobile host, which may be in its home network, indicated via channel $ls\_rep\_home$, or has moved outside, indicated via channel $ls\_rep\_out$.

- $DT4$ receives and acknowledges those datagrams destined to the home address of the mobile host($ip$). The error No.4 indicates a datagram destined to other communicating host is routed to the mobile host.

$$
\begin{aligned}
DT4_{k,i}(ip) = pkt\_out_k(src, tgt, payload). \\
if\ (tgt = ip)\ then\ \overline{ack_i}.DT4_{k,i}(ip) \\
else\ \overline{stderr}\langle 4\rangle.DT4_{k,i}(ip)
\end{aligned}
$$

For a mobile host, two local variables are defined, namely the movement flag($state$) and the last pending request($ip, coa$). The movement flag 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 to an incoming ARP Request message. $LS$ takes control of read/write accesses to $state$, which is initialized as $false$.

$$LS_k(state) = set\_ls_k(nstate).LS_k(nstate)$$
$$+ls\_req_k.$$
$$(if\ (state)\ then\ \overline{ls\_rep\_out}_k.LS_k(state)$$
$$else\ \overline{ls\_rep\_home}_k.LS_k(state))$$

The last pending request is the last Registration Request messages sent from the mobile host. $PR$ takes control of read/write accesses to $(ip, coa)$, in which the last COA advertised is initialized as none. The channel $set\_ls$ is used to set the movement flag according to the latest Registration Reply message.

$$PR_k(ip) = pending\_req_k(coa, state).PR'_k(ip, coa, state)$$
$$+is\_last_k(ha, coa).\overline{is\_last\_no}_k.PR_k(ip)$$
$$PR'_k(ip, coa, state) = pending\_req_k(ncoa, nstate).PR'_k(ip, ncoa, nstate)$$
$$+is\_last_k(ha, ncoa).$$
$$(if\ (ha = ip\ and\ ncoa = coa)\ then$$
$$\overline{is\_last\_yes}_k.\overline{set\_ls}_k\langle state\rangle.PR_k(ip)$$
$$else\ \overline{is\_last\_no}_k.PR'_k(ip, coa, state)$$

Note that the COA of a 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

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

### *3.1.2 Model Checking*

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. As required by the model checker we used, the property of deadlock freedom can be described as a $\mu$-calculus formula

$$DF \triangleq AG(\langle - \rangle\ true)$$

where $AG(\varphi) \equiv \nu X.\varphi\ \wedge\ [-]X.\ AG(\varphi)$ means the property $\varphi$ should always hold. In this way, $DF$ means the model should evolve forever.

The graphical user interface(GUI) for model checking $MIP4^{1,1,1}$ and its components is shown in Fig.1, where the message task component $AA_0$ is being verified against $DF$. The text window in its right side shows the result of model checking, as well as the corresponding statistics on time consumption.

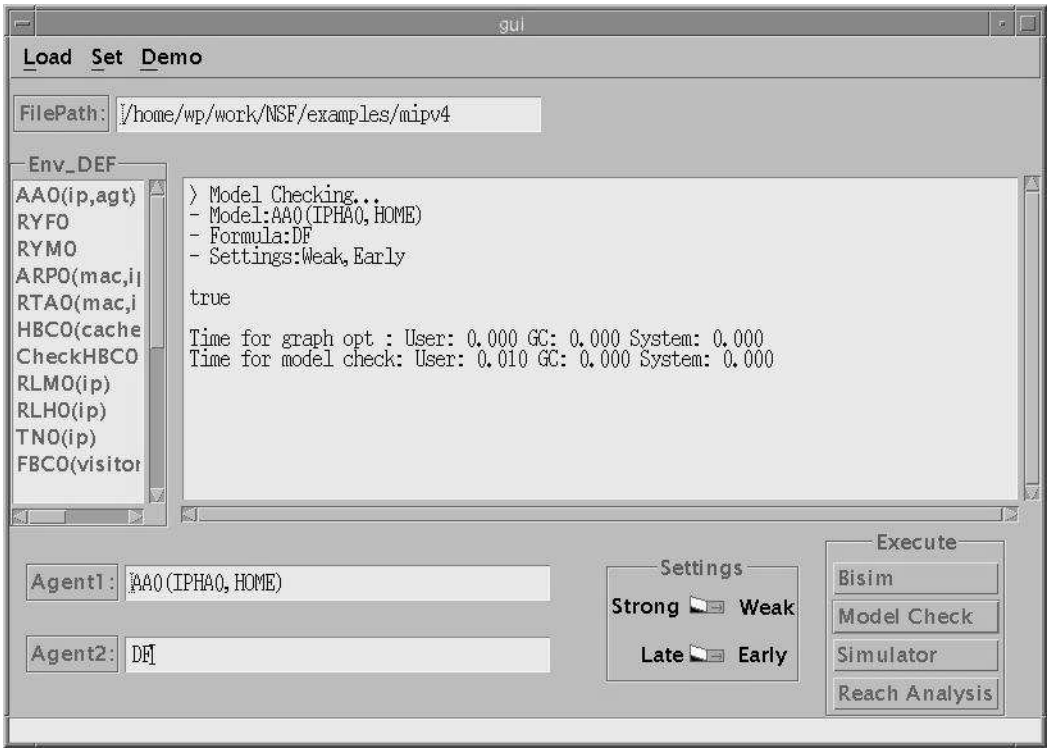


Fig. 1. Model Checking $MIP4^{1,1,1}$

Although $MIP4^{1,1,1}$ seems rather simple, the model checking experiment shows that it is far more complicated than can be dealt with by the model checker. However, the model checker can verify the deadlock freedom of its each task component. Table 2 illustrates the results of model checking these components 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. With Theorem 2.3 and 2.4 the deadlock freedom of $MIP4^{1,1,1}$ can be concluded without explicitly enumerating its state space exhaustively.

### *3.2 Mobile IPv6*

Mobile IPv6 simplifies Mobile IPv4 in that there is no need to deploy special routers as "foreign agents". It can operate in any location without any special support required from the local router. The differences between Mobile IPv6 and Mobile IPv4 include:

**Mobility Detection** An IPv6-based mobile host uses the Neighbor Discovery Protocol to locate itself, instead of specific agent discovery mechanism in Mobile IPv4.

**Registration** An IPv6-based mobile host sends directly registration requests to and receives registration replies from its home agent.

| **Task** | $\|S\|$ | $\|N\|$ | $\|E\|$ | $DF$ | **Task** | $\|S\|$ | $\|N\|$ | $\|E\|$ | $DF$ |
|---|---|---|---|---|---|---|---|---|---|
| $AA_0$ | 5 | 1 | 2 | TRUE | $FBC_0$ | 300 | 4 | 11 | TRUE |
| $RYF_0$ | 395 | 3 | 5 | TRUE | $AD_0$ | 330 | 6 | 17 | TRUE |
| $RYM_0$ | 135 | 3 | 5 | TRUE | $RRF_0$ | 265 | 4 | 7 | TRUE |
| $ARP_0$ | 40 | 4 | 6 | TRUE | $RRH_0$ | 265 | 4 | 7 | TRUE |
| $RTA_0$ | 252 | 7 | 10 | TRUE | $DT4_0$ | 30 | 2 | 3 | TRUE |
| $HBC_0$ | 434 | 7 | 14 | TRUE | $MARP_0$ | 20 | 4 | 5 | TRUE |
| $RLM_0$ | 134 | 3 | 5 | TRUE | $LS_0$ | 28 | 3 | 6 | TRUE |
| $RLH_0$ | 278 | 5 | 9 | TRUE | $PR_0$ | 1174 | 8 | 14 | TRUE |
| $TN_0$ | 179 | 6 | 9 | TRUE | | | | | |

Table 2
Results of Model Checking Each Component

**Routing** Datagrams destined to a roaming mobile host will be tunnelled directly to it.

Note that the terms of "Registration Request" and "Registration Reply" in Mobile IPv4 are referred to as "Binding Update" and "Binding Acknowledgement" in Mobile IPv6, respectively.

### *3.2.1 Modelling*

This section will illustrate a STGA model of Mobile IPv6 in the context of the modelling framework proposed in Section 2. Formally, Mobile IPv6 can be described as a tuple

$$MIP6^{n_{ha},n_{mh}} = (\mathbb{E}_6, \mathbb{C}_6^e, \mathbb{C}_6^p)$$

with

$$\mathbb{E}_6 = \{HA6[0..n_{ha}-1], MH6[0..n_{mh}-1]\}$$

where

(i) $HA6$ and $MH6$ are STGA models of home agents and mobile hosts in Mobile IPv6, respectively;

(ii) $n_{ha}$ and $n_{mh}$ are the number of home agents and mobile hosts, respectively;

Let $i, k$ range over indexes of home agents and mobile hosts, respectively. Table 3 describes how these entities are connected with the environment(denoted by -) via channels in $\mathbb{C}_6^e$(row 1 to 4) and how they are connected with each other via channels in $\mathbb{C}_6^p$(row 5 to 8). Suppose $N'_{mh} = \{k \mid 0 \leq k \leq n_{mh}-1\}$, $N'_{ha} = \{i \mid 0 \leq i \leq n_{ha}-1\}$, $hn : N'_{mh} \rightarrow N'_{ha}$ is a home network function

| Channel | From | To | Description |
|---|---|---|---|
| $send_i$ | - | $HA6_i$ | To receive the datagrams from the environment. |
| $location_k$ | $MH6_k$ | - | To indicate the location of the $(k+1)$-th mobile host. |
| $route_k$ | $HA6_{hn(k)}, MH6_k$ | - | To indicate the routing path for a datagram addressed to $(k+1)$-th mobile host. |
| $stderr$ | $HA6, MH6$ | - | To indicate an exception. |
| $reg\_req\_hm_i$ | $MH6$ | $HA6_i$ | To send a registration request directly to the $(i+1)$-th home agent. |
| $reg\_rep\_hm_k$ | $HA6$ | $MH6_k$ | To send a registration reply directly to the $(k+1)$-th mobile host. |
| $forward_k$ | $HA6$ | $MH6_k$ | To receive a datagram from a home agent. |
| $tunnel_j$ | $HA6$ | $MH6_j$ | To receive a tunnelled datagram from a home agent. |

Table 3
Channels in $\mathbb{C}_6^e \cup \mathbb{C}_6^p$

that maps each mobile host $MH6_k$ to its home agent $HA6_{hn(k)}$. Herein all channels in $\mathbb{C}_6^e \cup \mathbb{C}_6^p$ 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. Local channels in each entity are just as their names imply.

$MIP6$ can be easily proved to be interoperable so the deadlock freedom can be preserved.

**Home Agent**

A home agent contains three message task components: $RYM$, $ARP$ and $RTA$, which have been defined in Section 3.1.1.

Similarly the task component $HBC$ takes control of read/write accesses to the binding list($binding\_list$). Then, the home agent can be formally described as

$$HA6_i(mac, ip, binding\_list) = RYM_i \mid \prod_{hn(k)=i} ARP_k \mid RTA_i(mac, ip) \mid HBC_i(binding\_list)$$

**Mobile Host**

A mobile host contains four message task components: $ND$, $RRH$, $MARP$ and $DT6$. $RRH$ and $MARP$ have been defined in Section 3.1.1.

- Movement detection in IPv6 was considered as an isolated protocol, which is not modelled in this case study. Once detecting the movement, $ND$ will inform the home agent of the mobile host to update its binding status with a care-of address in a Binding Update message.

$$\begin{aligned} ND_{k,i}(ip, hip, coa) = {} & \overline{pending\_req_k}\langle 0, false\rangle. \\ & \overline{reg\_req\_hm_i}\langle ip, 0, k\rangle.ND_{k,i}(ip, hip, coa) \\ & +\overline{pending\_req_k}\langle icoa, true\rangle. \\ & \overline{reg\_req\_ha_j}\langle ip, icoa, k\rangle.ND_{k,i}(ip, hip, coa) \\ & +set\_careof_k(ncoa). \\ & \overline{\mathbf{location_k}}\langle \mathbf{ncoa}\rangle.ND_{k,i}(ip, hip, ncoa) \end{aligned}$$

- $DT6$ receives and acknowledges those datagrams destined to the home address of the mobile host($ip$). Those datagrams may be forwarded via the channel $pkt\_out$ when the mobile host is in its home network, or tunnelled via the channel $tu\_ha$ when it is out of the home network.

$$\begin{aligned} DT6_{k,i}(ip) = {} & pkt\_out_k(src, tgt, payload). \\ & if\ (tgt = ip)\ then\ \overline{ack_i}.DT6_{k,i}(ip) \\ & else\ \overline{stderr}\langle 4\rangle.DT6_{k,i}(ip) \\ & +tu\_ha_k(tsrc, ttgt, src, tgt, payload).\overline{ls\_req_k}. \\ & (ls\_rep\_out_k.\overline{\mathbf{route_k}}\langle \mathbf{BYFOREIGN}\rangle.\overline{ack_i}.DT6_{k,i}(ip) \\ & +ls\_rep\_home_k.\overline{stderr}\langle 3\rangle.DT6_{k,i}(ip) \end{aligned}$$

Similarly $LS$ and $PR$ takes control of read/write accesses to the movement flag($state$) and the last pending request($ip, coa$), respectively. Then the mobile host can be formally described as

$$\begin{aligned} MH6_k(ip, hip) = {} & ND_{k,hn(k)}(ip, hip, 0) \mid RRH_k \mid MARP_k(mac) \\ & \mid DT6_{k,hn(k)}(ip) \mid LS_k(false) \mid PR_k(ip) \end{aligned}$$

### *3.2.2 Model Checking*

A representative instance of $MIP6$ is considered in the case study, i.e. $MIP6^{1,1}$ that contains only one home agent and one mobile host. From the viewpoint of functionality, Mobile IPv6 should always be able to route IP datagrams to mobile nodes roaming outside of the home networks. As required by the model checker we used, this property can be described as the conjunction of the following $\mu$-calculus formulae:

(i) Deadlock Freedom($DF$)

$$DF \triangleq AG(\langle - \rangle\ true)$$

(ii) Adaptive Routing($AR$)

$$AR_h \triangleq AG([ah]AG([s]EF(\langle rh \rangle true - af)))$$
$$AR_f \triangleq AG([af]AG([s]EF(\langle rf \rangle true - ah)))$$

(iii) Tunnel on Demand($ToD$)

$$ToD \triangleq AG([\overline{stderr}\langle 3 \rangle]\ false)$$

where

(i) $s \equiv send(imac, src, tgt, payload)$, an input action to retrieve a datagram from the environment;

(ii) $ah \equiv \overline{location}\langle HOME \rangle$, an output action to indicate the mobile host is home now;

(iii) $af \equiv \overline{location}\langle FOREIGN \rangle$, an output action to indicate the mobile host has moved out;

(iv) $rh \equiv \overline{route}\langle HOME \rangle$), an output action to indicate a datagram is routed directly to its destination without passing a tunnel;

(v) $rh \equiv \overline{route}\langle FOREIGN \rangle$, an output action to indicate a datagram is routed to its destination via a tunnel;

(vi) $EF(\varphi - \alpha) \equiv \mu X.\varphi \vee [-\alpha]X$, which means the property $\varphi$ should hold eventually except when action $\alpha$ has taken place.

Recall that $DF$ means the model should always evolve. $AR_h(AR_f)$ means whenever the mobile host becomes stable at the home(foreign) network, all datagrams destined for the mobile host should be forwarded without passing a tunnel(via a tunnel). $ToD$ means a tunnel should be used only when the mobile host is roaming.

The graphical user interface for model checking $MIP6^{1,1}$ and its components is shown in Fig.2, where the message task component $RYM_0$ is being verified against $DF$.

Although $MIP6^{1,1}$ seems rather simple, the model checking experiment shows that it is far more complicated than can be dealt with by the model checker. However, the model checker can verify the deadlock freedom of its each task component. Table 4 illustrates the results of model checking these components against $DF$. With Theorem 2.3 and 2.4 the deadlock freedom of $MIP6^{1,1}$ can be concluded without explicitly enumerating its state space exhaustively.

As far as $AR$ and $ToD$ are concerned, the model checking experiments end unexpectedly with negative results, which disclose the embarrassing infrangibility of Mobile IPv6 in its routing capability. Two ways to implement a home agent are verified with the only difference in the order between replying a mobile host with a Binding Acknowledgement and updating its local binding list:

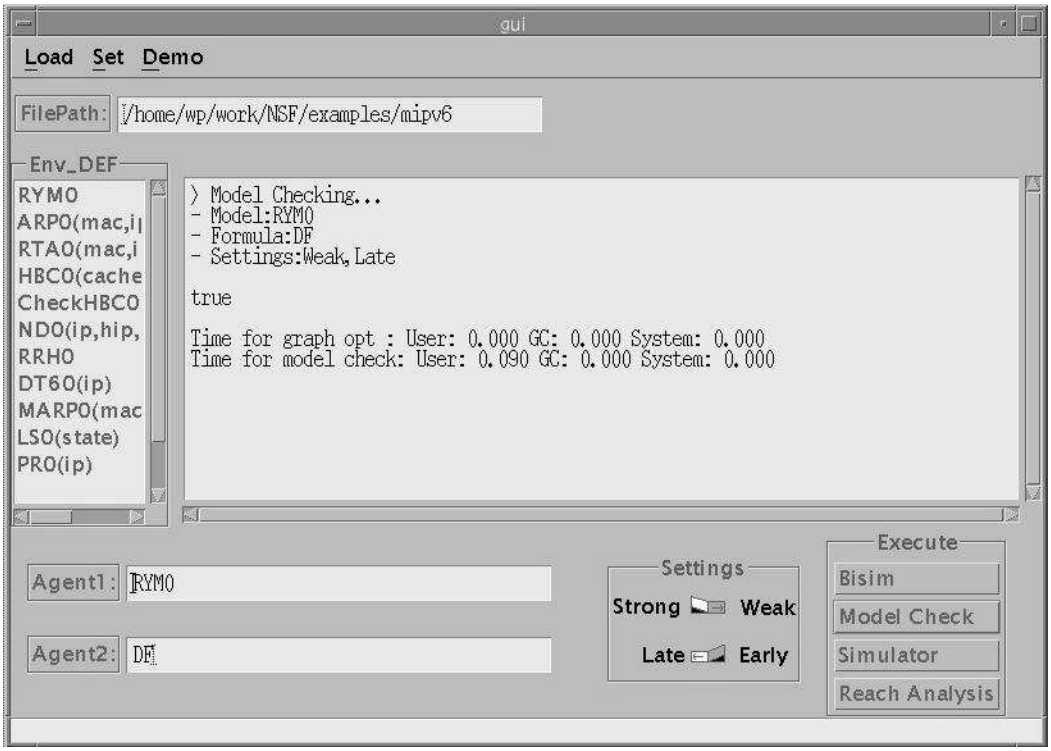


Fig. 2. Model Checking $MIP6^{1,1}$

| **Task** | $\|S\|$ | $\|N\|$ | $\|E\|$ | $DF$ |
|---|---|---|---|---|
| $RYM_0$ | 135 | 3 | 5 | TRUE |
| $ARP_0$ | 40 | 4 | 6 | TRUE |
| $RTA_0$ | 252 | 7 | 10 | TRUE |
| $HBC_0$ | 434 | 7 | 14 | TRUE |
| $ND_0$ | 117 | 5 | 9 | TRUE |
| $RRH_0$ | 265 | 4 | 7 | TRUE |
| $DT6_0$ | 151 | 6 | 11 | TRUE |
| $MARP_0$ | 20 | 4 | 5 | TRUE |
| $LS_0$ | 28 | 3 | 6 | TRUE |
| $PR_0$ | 1174 | 8 | 14 | TRUE |

Table 4
Results of Model Checking Each Component

one way is to reply first, then update; while the other, vice versa. Although familiar in known implementations of IPv6 [13], these two ways do not satisfy properties $AR$ and $ToD$.

(i) If a home agent replies a mobile host before it updates its local repository, then all datagrams received during the period of these two events will be forwarded to an outdated address, which is unreachable on behalf of the mobile host;

(ii) If a home agent replies a mobile host after it has updated its local repository, then during the period of these two events, the mobile host is unstable in the sense that its roaming capability has not yet been enabled. Any datagram received during this period will be forwarded to the mobile host, of which the behavior is undefined in Mobile IPv6.

By model checking $MIP6^{1,1}$ directly, the counterexamples generated from the model checker definitely clarify that these infrangibilities are resulted from the binding incoherency during mobile communications, which has also not been reflected by Mobile IPv6 testing [17].

**Adaptive Routing**

Fig.3 illustrates a counterexample for the property $AR_f$, where a mobile host is to register its binding information. Although having acknowledged the mobile host with $OK$, the home agent updates its local binding list after it has forwarded a datagram received previously to the mobile host according to its current binding list, where the mobile host is supposed to be still home. Therefore the datagram will never reach the mobile host.

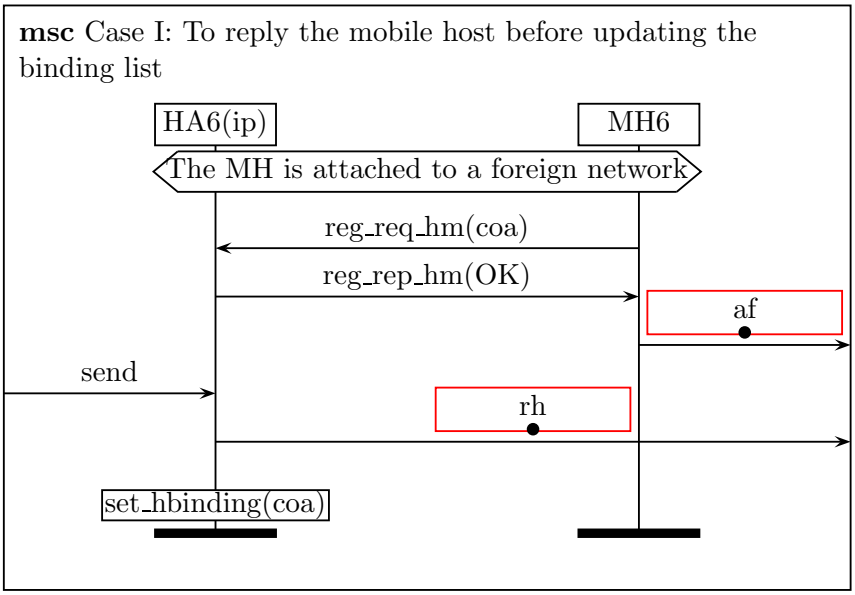


Fig. 3. Counterexample for $AR_f$

Similarly one can find a corresponding counterexample for the property $AR_h$ with respect to $MIP6^{1,1}$.

One way to avoid the case of datagram loss above is to switch the order between actions *reg_rep_hm* and *set_hbinding*. A model checking experiment shows the resulted model does still not satisfy $AR_h$ with a counterexample illustrated in Fig.4, where a mobile host is to move out of its home network. Before the mobile host becomes stable at the foreign network, that is, receives a Binding Acknowledgement from its home agent, a datagram has been forwarded to the new care-of address that is not yet enabled. In such case, the behavior of the mobile host is undefined in the specification of Mobile IPv6.

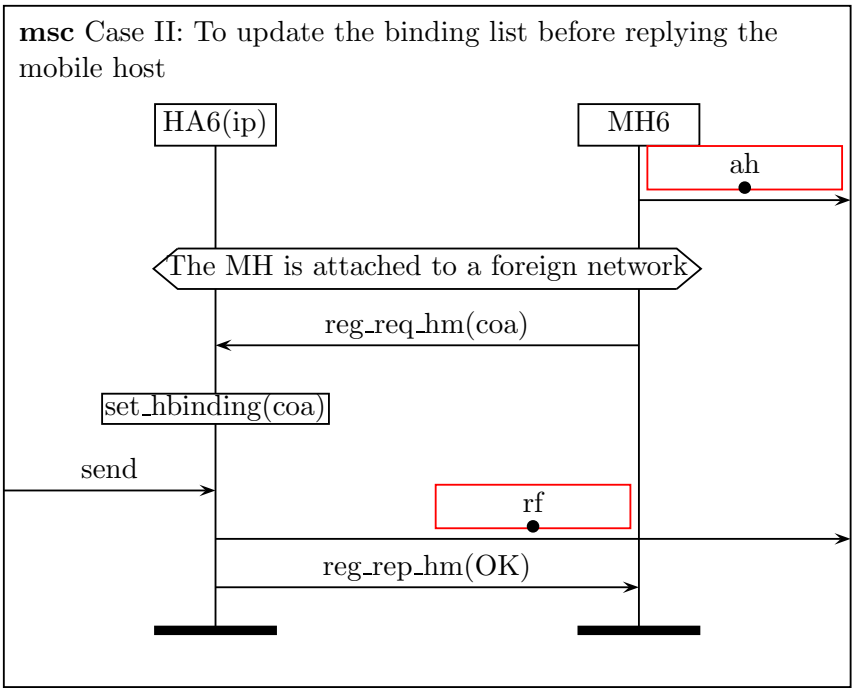


Fig. 4. Counterexample for $AR_h$

Similarly one can find a corresponding counterexample for the property $AR_f$ with respect to the modified model.

**Tunnel on Demand**

The binding incoherency also ruins the property $ToD$. Fig.5 illustrates a counterexample for $ToD$, where a mobile host is to deregister its binding information. Although having acknowledged the mobile host with $OK$, the home agent updates its local binding list after it has forwarded a datagram received previously to the mobile host according to its current binding list, where the mobile host is supposed to be out of its home network. In such case, the behavior of the mobile host is undefined in the specification of Mobile IPv6.

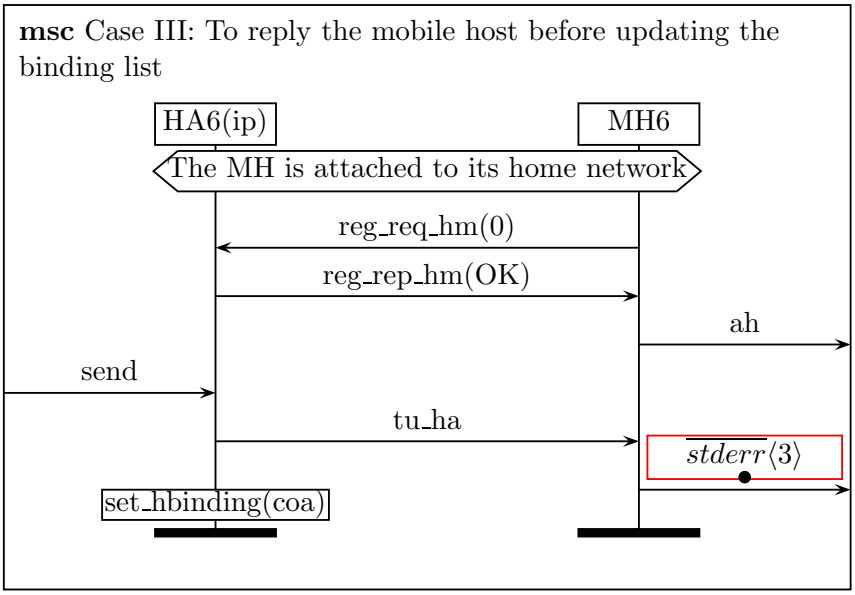


Fig. 5. Counterexample for $ToD$

The model resulted by switching the order between actions *reg_rep_hm* and *set_hbinding* does still not satisfy $ToD$. A counterexample is illustrated in Fig.6, where a mobile host is to register its binding information. Before receiving a Binding Acknowledgement from its home agent, it receives a datagram via a tunnel, which has not yet been enabled. In such case, the behavior of the mobile host is undefined in the specification of Mobile IPv6.

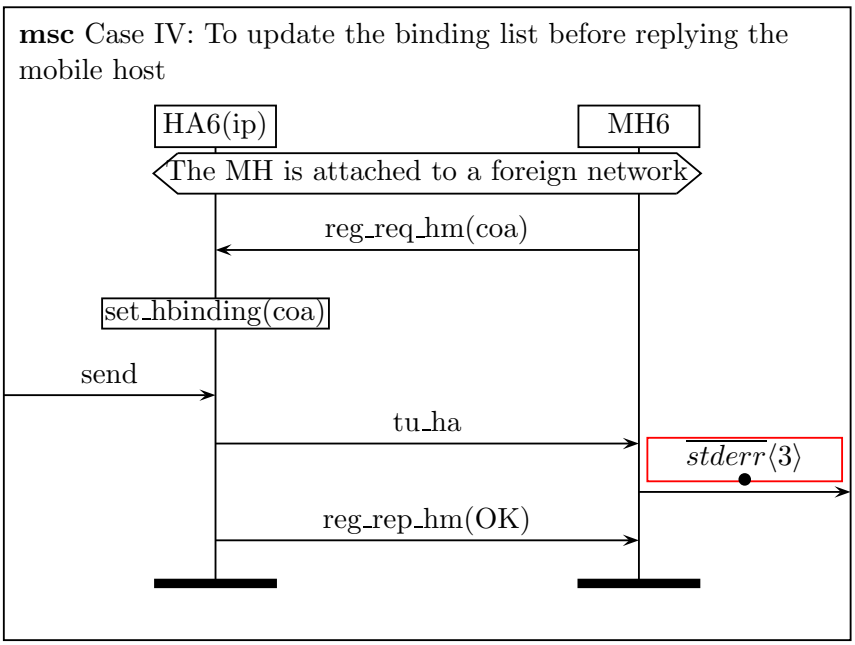


Fig. 6. Another Counterexample for $ToD$

## 4 Conclusion

A symbolic and compositional framework was presented in this paper presents 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 communicating with each other via message passing strategy. Case studies with Mobile IPv4 and IPv6 illustrate the effectiveness of the modelling framework. The main advantages of the framework rest on the following aspects:

(i) It can address dynamic network topologies without additional syntactic or pragmatic mobility facilities.

(ii) It preserves the deadlock freedom in the sense that the deadlock freedom of a 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.

(iii) The case study with Mobile IPv6 detects the infrangibility of Mobile IPv6 in its binding incoherency, which may make a datagram unreachable to its destination or being forwarded to an unstable mobile host. The behaviors of home agents and mobile hosts in such cases are not addressed in the specification of Mobile IPv6 and even not touched in Mobile IPv6 testing.

As future work, we would like to analyze other critical properties, e.g. mutual exclusion from a compositional perspective. Moreover, 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] Amadio, R. M. and S. Prasad, *Modelling IP mobility*, in: D. Sangiorgi and R. de Simone, editors, *Ninth International Conference on Concurrency Theory*, Lecture Notes in Computer Science 1466 (1998), pp. 301–316.

[2] Dang, Z. and R. A. Kemmerer, *Using the ASTRAL model checker to analyze mobile IP*, in: *21st international conference on Software engineering* (1999), pp. 132–141.

[3] Dong, Y., X. Du, G. J. Holzmann and S. A. Smolka, *Fighting livelock in the GNU i-protocol: A case study in explicit-state model checking*, International Journal on Software Tools for Technology Transfer **4** (2003), pp. 505–528.

[4] Holzmann, G. J., "Design and Validation of Computer Protocols," Prentice Hall, 1991.

[5] Johnson, D. and C. Perkins, *Mobility support in IPv6*, in: *Second*

*ACM International Conference On Mobile Computing And Networking (MobiCom'96)*, ACM, 1996, pp. 27–37.

[6] Johnson, D., C. Perkins and J. Arkko, *Mobility support in IPv6*, Internet-Draft, IETF Mobile IP Working Group (2003).

[7] Lin, H., *Symbolic transition graph with assignment*, in: *CONCUR'96*, Lecture Notes in Computer Science **1119**, Pisa, Italy, 1996, pp. 50–65.

[8] Lin, H., *Model checking value-passing processes*, in: *8th Asia-Pacific Software Engineering Conference (APSEC'2001)*, Macau, 2001, pp. 3–12.

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

[10] Musuvathi, M. and D. Engler, *Model checking large network protocol implementations*, in: *The First Symposium on Networked Systems Design and Implementation*, USENIX Association, 2004, pp. 155–168.

[11] Perkins, C., *IP mobility support*, IETF RFC 2002 (1996).

[12] Perkins, C., *IP mobility support for IPv4*, IETF RFC 3344 (2002).

[13] Santti, K., S. Auvray and G. Egeland, *Survey of mobile IPv6 implementations*, Tsunami Project P1113, European Institute for Research and Strategic Studies in Telecommunications(EURESCOM) (2002).

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

[15] Wu, P., *Analyzing interoperability of protocols using model checking* (2003), accepted by the 1st International Workshop on Automated Technology for Verification and Analysis (ATVA'03), Taipei, Taiwan.

[16] Wu, P. and D. Zhang, *Compositional analysis of mobile IP with symbolic transition graphs*, in: *16th International Conference on Computer Communication*, 2004, pp. 1481–1488.

[17] Yu-jun, Z., T. Jun and L. Zhong-cheng, *Mobile IPv6 and its conformance testing* (2002).

[18] Zhang, Y., "Extending a Model-Checking Tool with Non-trivial Data Structures," Master's thesis, Institute of Software, Chinese Academy of Sciences (2003).