Formal Modeling of Mobile Middleware for Tuple Space Coordination over Multiple Heterogeneous Networks

Suddhasil De, Diganta Goswami, and Sukumar Nandi

Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati, Assam – 781039, India
{suddhasil,dgoswami,sukumar}@iitg.ernet.in

Abstract. Tuple Space based Mobile Middleware (TSMM), with tuple space as coordination medium, exhibits multiple decoupling qualities during coordination, which enhances its robustness and flexibility and makes it an appropriate coordination platform for underlying mobile and dynamic networks. However, formal semantics of TSMM are required for reasoning TSMM as coordination platform, which also help in developing supported applications. This paper suggests an approach of formalizing TSMM that can be deployed over multiple heterogeneous mobile and dynamic networks. Formalization is carried out using Mobile UNITY.

Keywords: Mobile middleware, coordination, tuple space, robustness, formalization, Mobile UNITY.

1 Introduction

Advances in wireless communication technologies and mobile computing devices lead to the deployment of different scales of wireless networks. These networks are characterized by device mobility, network dynamics, and inherent unreliability in communication links. They are targeting different types of applications for the benefits of end users. Most of their applications require coordination support for proper functioning to achieve a common goal. Services of *mobile middleware* [1], with proper *coordination medium* incorporated within it, becomes inevitable for facilitating coordination among different active components of a supported application (called *agents*) executing in computing environments of different devices (called *hosts*). One such coordination medium, tuple space [2], attains different dimensions of uncoupling between interacting agents [3], even in underlying heterogeneous networks. Mobile middleware incorporating tuple space for coordination is referred as *Tuple Space based Mobile Middleware* (TSMM) [4].

In TSMM, *tuple* is basic unit of data exchanged during agent interactions via a shared repository (called *tuple space*), while *antituple* is basic unit of search key to identify tuples residing in tuple space. Tuple space as coordination medium uncouples interacting agents about time, space (i.e. naming), interacting data and operations on them [3,5]. Uncoupling in several dimensions

K. Singh, A.K. Awasthi, and R. Mishra (Eds.): QSHINE 2013, LNICST 115, pp. 415–430, 2013.

enable TSMM of providing loose coupling of coordination, which enhances its robustness and flexibility and makes it an appropriate coordination platform for underlying dynamic heterogeneous networks. However, to facilitate application designers, formal specifications of semantics of TSMM are required to be clearly stated. Formalization not only enables proper analysis of robustness and flexibility of TSMM over heterogeneous networks, but also defines its precise semantics and prepares foundation for its implementation. This paper extends an earlier work [6], by providing a formal treatment to TSMM that is deployed over multiple mobile, dynamic and unreliable heterogeneous networks. Mobile UNITY [7], a general-purpose reasoning tool, is used for formalization.

In literature, formal semantics of tuple space model has been presented earlier [5,8,9]. In these works, basic tuple space operations and agent mobility of TSMM are formalized using Mobile UNITY. However, unlike [5,8,9], this paper focuses on formalizing aggregated functionalities of TSMM, including multiple dimensions of uncoupling, communication and discovery mechanisms etc. Tuple space operations are abstracted in this formalization as simple calls to respective primitives, while agent mobility is abstracted as a function to simplify its representation. Also, this paper shows building of formal representation of TSMM by combining individual specifications of its different functionalities. Compared to [6], this paper extends by including support of multiple underlying heterogeneous networks, all of which are mobile, dynamic and unreliable. In particular, TSMM, formalized in [6], considers Infrastructure Basic Service Set (iBSS) [10] as the only underlying network, whereas, TSMM in this paper supports both iBSS and Independent Basic Service Set (IBSS) [10]. Two heterogeneous networks are considered for this paper to keep the formalization readable. However, this formal treatment can be easily extended to include other underlying heterogeneous networks for TSMM. Rest of the paper is organized as follows. Section 2 gives a brief overview of TSMM, which is next formalized using Mobile UNITY in Section 3. Finally, Section 4 concludes the paper.

2 Overview of TSMM Having Multiple Decoupled Coordination

TSMM is the coordination platform to support agent interactions in mobile distributed applications, thereby providing ubiquity to user activities.

Architecture. TSMM comprise of several components, which can be organized within agent or host. Each instance of agent contains an *agent tuple space* (`ATS`) and its interfaces, local operation manager, remote operation manager, ATS reaction manager and acquaintance list. One instance of host runs in one device and supports execution of single/multiple agents. In each host, different components manage functionalities of communication, discovery, host server, *host tuple space* (`HTS`) and its interfaces, agent management, mobility etc. Architecture of TSMM with all its components is shown in figure 1.

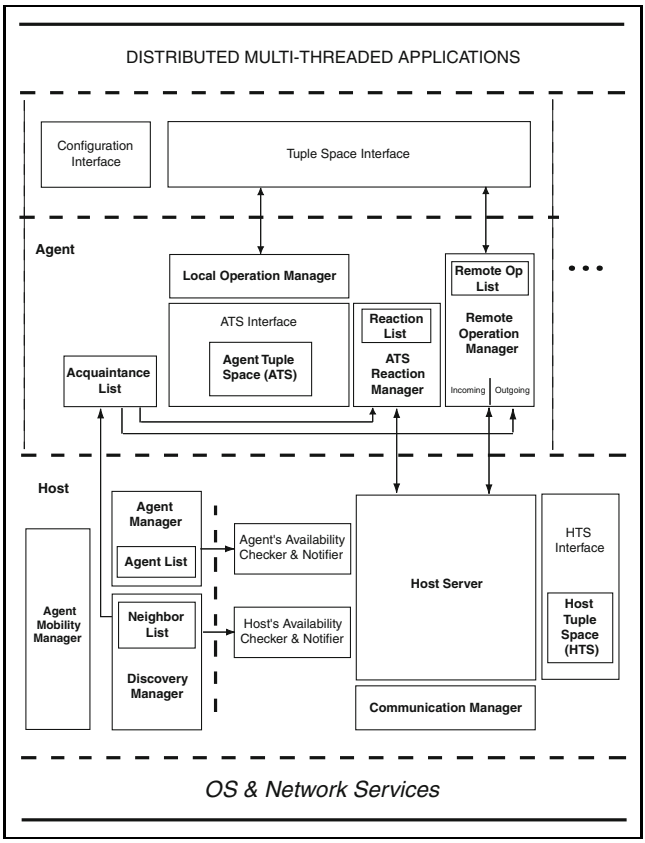

Fig. 1. Architecture of TSMM showing its significant components [6]

Tuple Space Model. In TSMM, tuples and antituples comprise of *unordered* sequence of fields [11], whereas tuple space is *indexed* in nature [12]. During interaction between any pair of agents, initiator of interaction becomes *reference agent* and destination becomes *target agent*. Different primitives are defined for writing, reading and withdrawing tuples from tuple space (like, `ATS`) using tuple-producing, tuple-reading and tuple-consuming primitives. Tuple-producing primitives cover `out` and `outg`, while tuple-reading primitives include `rd`, `rdp`, `rdg` and `rdgp`, and tuple-consuming primitives are `in`, `inp`, `ing` and `ingp`, details of which are given in [5]. Local primitives are executed in own `ATS`, whereas for executing remote primitives, invoked parameters are shipped to specified target agent(s), executed in its `ATS` and results of execution are sent back.

Reactivity Model. TSMM incorporates *reactivity* in `ATS` to monitor and respond to different events (like, presence of a particular tuple in tuple space etc.) during execution. Reactivity is implemented by generating and registering *reaction* in `ATS`. Registered reaction, with condition specified by antituple, fires if

that condition gets satisfied i.e. antituple matches tuple in tuple space. Firing of reaction signifies execution of application-defined reactive codes, like notifying presence of tuples, withdrawing tuples from `ATS` etc, and responses are sent back.

Decoupled Coordination Model. In TSMM, agent interactions use decoupled reactivity [5], whereby `HTS` provides additional decoupling medium to accomplish complete decoupling of agent interactions. `HTS` stores two special tuples (viz. reaction tuple and response tuple). Reaction tuples are created from different parameters of invoked remote primitives, while response tuples are created from result of execution of these primitives. Reaction tuple is first inserted into `HTS` of reference host. On availability of target host, it is withdrawn from that `HTS`, passed through underlying infrastructure to target host, and subsequently inserted into its `HTS`. Eventually, reaction tuple is withdrawn from target host's `HTS`, once target agent is available. Parameters of invoked primitive is next extracted and execution of that primitive starts at `ATS` of target agent. In case of remote tuple-reading and -consuming primitives, target agent packs results of execution (viz. sought tuple(s) from its `ATS`) into response tuple. Following previous approach, that response tuple eventually reaches reference agent, and sought tuple(s) are extracted from it. For achieving consistency in coordination, reference agent responds back to target agent(s) with additional ACK tuple and NACK tuple for any invoked remote tuple-consuming primitive. ACK tuple positively acknowledges acceptance of responded tuple as sought tuple, whereas NACK tuple returns non-accepted responded tuple back.

Supplementary Components. For execution over multiple unreliable networks having mobility, as well as for resolving heterogeneity of such multiple underlying networks, TSMM includes its own communication and discovery mechanisms [13] that uses transport service for data transmission. This paper considers iBSS and IBSS as underlying networks for TSMM. When deployed over iBSS, three categories of hosts are earmarked, viz. *stationary host*, *mobile host* and *access point*, whereas for IBSS, deployed hosts are earmarked as *mobile host* only. Discovery mechanism furnishes an updated knowledge of available agents and hosts. This knowledge is attained by sending and receiving beacons and is preserved in `NeighborList`. Communication mechanism emphasizes on reliably transferring reaction/response tuples from one host to another. It uses additional acknowledgement mechanism to achieve this reliability.

3 Proposed Approach of Formalization of TSMM

This section proposes an approach of formalizing TSMM as a Mobile UNITY system, comprising of a set of formal programs representing different agents and hosts. Favoring Mobile UNITY over other formal tools is due to its suitability in formalizing inherently non-terminating programs (like mobile middleware) and reasoning about agents temporal behavior using its proof rules. **System** *TSMM*, shown in Figure 2, comprises of multiple instances of two Mobile UNITY programs, and their interactions are specified in **Interactions** section.

System *TSMM*

Program $host(i)$ **at** λ

$\vdots$ $\vdots$ {Program description of $host(i)$, given separately}

Program $agent(k)$ **at** λ

$\vdots$ $\vdots$ {Program description of $agent(k)$, given separately}

Components

$\langle [] \; i :: host(i) \; \rangle \; [] \; \langle [] \; k :: agent(k) \; \rangle$

Interactions

{Attach $\mathcal{T}_{\text{W}}$ of hosts with wired network interfaces in $iBSS$ as transiently-shared variable when connected}

$shared_{\text{W}_{\text{iBSS}}}$ ::

$\langle [] \; i, j :: host(i).\mathcal{T}_{\text{W}} \approx host(j).\mathcal{T}_{\text{W}}$

when $(host(i).nwdeploy = iBSS) \wedge (host(j).nwdeploy = iBSS) \wedge (host(i)\Gamma' host(j))$
$\wedge \big(\texttt{isSH}(host(i)) \vee \texttt{isAP}(host(i))\big) \wedge \big(\texttt{isSH}(host(j)) \vee \texttt{isAP}(host(j))\big)$

engage $host(i).\mathcal{T}_{\text{W}}$ disengage current $\| \bot \; \rangle$

{Attach $\mathcal{T}_{\text{WL}}$ of mobile host and access point in $iBSS$ as transiently-shared variable, only when colocated}

$[] \; shared_{\text{WL}_{\text{iBSS}}}$ ::

$\langle [] \; i, j :: host(i).\mathcal{T}_{\text{WL}} \approx host(j).\mathcal{T}_{\text{WL}}$

when $(host(i).nwdeploy = iBSS) \wedge (host(j).nwdeploy = iBSS) \wedge (host(i)\Gamma' host(j))$
$\wedge \big(\big(\texttt{isMH}(host(i)) \wedge \texttt{isAP}(host(j))\big) \vee \big(\texttt{isAP}(host(i)) \wedge \texttt{isMH}(host(j))\big)\big)$

engage $host(i).\mathcal{T}_{\text{WL}}$ disengage current $\| \bot \; \rangle$

{Attach $\mathcal{T}_{\text{WL}}$ of mobile hosts in $IBSS$ as transiently-shared variable, only when colocated}

$[] \; shared_{\text{WL}_{\text{IBSS}}}$ ::

$\langle [] \; i, j :: host(i).\mathcal{T}_{\text{WL}} \approx host(j).\mathcal{T}_{\text{WL}}$

when $(host(i).nwdeploy = IBSS) \wedge (host(j).nwdeploy = IBSS) \wedge (host(i)\Gamma' host(j))$
$\wedge \texttt{isMH}(host(i)) \wedge \texttt{isMH}(host(j))$

engage $host(i).\mathcal{T}_{\text{WL}}$ disengage current $\| \bot \; \rangle$

{Prepare to register active agents in respective hosts}

$[] \; regAgent :: \langle [] \; i, k :: host(i).\mathcal{Q}_{in} := host(i).\mathcal{Q}_{in} \bullet agent(k).aid$ when $(host(i).\lambda = agent(k).\lambda) \; \rangle$

{Prepare to deregister terminated/migrated agents from respective hosts}

$[] \; deregAgent :: \langle [] \; i, k :: host(i).\mathcal{Q}_{out} := host(i).\mathcal{Q}_{out} \bullet agent(k).aid$ when $\neg(host(i).\lambda = agent(k).\lambda) \; \rangle$

{Prepare to transfer reaction/response tuple from agent to host}

$[] \; \langle [] \; i, k :: host(i).\mathcal{Q}_{T^S_{a_k}}, agent(k).\mathcal{Q}_{T^S_{a_k}} := host(i).\mathcal{Q}_{T^S_{a_k}} \bullet \texttt{head}(agent(k).\mathcal{Q}_{T^S_{a_k}}), \texttt{tail}(agent(k).\mathcal{Q}_{T^S_{a_k}})$
when $(host(i).\lambda = agent(k).\lambda) \wedge \neg(agent(k).\mathcal{Q}_{T^S_{a_k}} = \bot) \; \rangle$

{Prepare to transfer reaction/response tuple from host to agent}

$[] \; \langle [] \; i, k :: agent(k).\mathcal{Q}_{T^R_{a_k}}, host(i).\mathcal{Q}_{T^R_{a_k}} := agent(k).\mathcal{Q}_{T^R_{a_k}} \bullet \texttt{head}(host(i).\mathcal{Q}_{T^R_{a_k}}), \texttt{tail}(host(i).\mathcal{Q}_{T^R_{a_k}})$
when $(host(i).\lambda = agent(k).\lambda) \wedge \neg(host(i).\mathcal{Q}_{T^R_{a_k}} = \bot) \; \rangle$

end

Fig. 2. Mobile UNITY system of TSMM

i-th host is specified by **Program** $host(i)$, whereas k-th agent is represented by **Program** $agent(k)$, where i and k are assumed to be quantified over appropriate ranges. Different conditions of interactions in **Interactions** section are enforced through **when** clauses. Clauses **engage** and **disengage**, and construct **current** are used for transient sharing between different hosts. Also, first three statements in **Interactions** section, labeled as $shared_{\text{W}_{\text{iBSS}}}$, $shared_{\text{WL}_{\text{iBSS}}}$ and $shared_{\text{WL}_{\text{IBSS}}}$, are

Program $agent(k)$ **at** λ

declare

$type$: $\in\{stationary, mobile\}$
[] $aid, taid, a$: agentid [] $taids$: sequence of agentid
[] T : tuple space [] $t, tuple$: tuple [] $\mathbf{t}, tuples$: set of tuple [] $\mathsf{a}, atuple$: antituple
[] r : RT_{tuple} [] $\mathcal{Q}_{T^S_{a_k}}, \mathcal{Q}_{T^R_{a_k}}$: queue of RT_{tuple}
[] $prid$: primitiveid [] $prType$: $\in\{local, remote\}$
[] ROL : sequence of (primitiveid, primitivename, set of agentid of target agents)
[] RL : sequence of (reactionid, primitiveid) [] $\mathbb{T}$: set of {agentid, set of tuple}
[] $prName$: $\in\{\mathsf{OUT}, \mathsf{OUTG}, \mathsf{RD}, \mathsf{RDG}, \mathsf{RDP}, \mathsf{RDGP}, \mathsf{IN}, \mathsf{ING}, \mathsf{INP}, \mathsf{INGP}\}$
[] $mode$: $\in\{\texttt{ONCE}, \texttt{ONCE/TUPLE}\}$
[] $TAs, rform$: natural
[] $prBulk, prRdIn, UsrRdy4Evt$: boolean

always

$aid := \texttt{getMyAgentID}(k)$ [] $type := \texttt{getAgentType}(stationary, mobile)$
[] $\text{isPresent}_{\text{in}ROL}(prid, taid) \equiv \langle\ \exists e :: (e \in ROL) \wedge (e \uparrow 1 = prid) \wedge (aid \in e \uparrow 3)\ \rangle$
[] $\text{isEmpty}_{\text{in}ROL}(prid) \equiv \langle\ \exists e :: (e \in ROL) \wedge (e \uparrow 1 = prid) \wedge (e \uparrow 3 = \emptyset)\ \rangle$

initially

$\lambda = \texttt{Location}(k)$
[] $TAs = 0$ [] $rform = 0$ [] $\mathsf{T} = \perp$ [] $ROL = \perp$ [] $RL = \perp$ [] $\mathbb{T} = \emptyset$
[] $t = \varepsilon$ [] $tuple = \varepsilon$ [] $\mathbf{t} = \emptyset$ [] $tuples = \emptyset$ [] $\mathsf{a} = \varepsilon$ [] $atuple = \varepsilon$
[] $\mathcal{Q}_{T^S_{a_k}} = \perp$ [] $\mathcal{Q}_{T^R_{a_k}} = \perp$ [] $UsrRdy4Evt = FALSE$

assign

{Migrate to different location}
[] $\lambda := \texttt{Location}(\texttt{Move}())$ if $(type = mobile)$

{Capture different parameters when user application is ready}
[] $\langle$ $prType, prName, UsrRdy4Evt := \texttt{getPrimType}(), \texttt{getPrimName}(), FALSE$
|| $prRdIn, prBulk := \texttt{getPrimRDorIN}(), \texttt{getPrimBulk}()$
|| $tuple := \texttt{getTuple}()$ if $\big((prRdIn = FALSE) \wedge (prBulk = FALSE)\big)$
|| $tuples := \texttt{getTuples}()$ if $\big((prRdIn = FALSE) \wedge (prBulk = TRUE)\big)$
|| $atuple := \texttt{getAntiTuple}()$ if $(prRdIn = TRUE)$
|| $TAs := \texttt{getTargetAgentCount}()$ if $(prType = remote)$
|| $\langle \| a : 1 \le a \le TAs :: taids[a] := \texttt{getTargetAgentID}(a) \rangle$ if $(prType = remote)$
|| $mode := \texttt{getMode}(\texttt{ONCE}, \texttt{ONCE/TUPLE})$ if $\big((prType = remote) \wedge (prRdIn = TRUE)\big)$
$\rangle$ if $(UsrRdy4Evt = TRUE)$

Fig. 3. Mobile UNITY **Program** $agent(k)$: part 1

reactive statements as they have used "≈" notation. Agent behaviors, including functionalities of ATS, Local Operation Manager, Remote Operation Manager, ATS Reaction Manager etc. are contained in $agent(k)$ as shown in Figure 3, Figure 4, and Figure 5. Similarly, functionalities of different components of host, including Transport Interface, Discovery Manager, Communication Manager, Host Server, Agent Manager etc., are contained in $host(i)$ as shown in Figure 6, Figure 7, Figure 8, and Figure 9. However, in above formal system, many aspects of TSMM are not directly formalized, to keep this formal system simple.

Different variables related to hosts and agents are specified in this formal system. For instance, $\mathcal{Q}$ is used to express any queue used to define different

{- - - - - - - - - - Start of Local Operation Manager - - - - - - - - - -}

{Perform different local tuple space primitives}

[] $\langle\ t, tuple, prType := tuple, \varepsilon, \varepsilon \parallel \texttt{out}(t, \mathbf{T})\ \rangle$ if $\big((prType = local) \wedge (prName = \mathsf{OUT}) \wedge \neg(tuple = \varepsilon)\big)$

[] $\langle\ \mathbf{t}, tuples, prType := tuples, \emptyset, \varepsilon \parallel \texttt{outg}(\mathbf{t}, \mathbf{T})$

$\rangle$ if $\big((prType = local) \wedge (prName = \mathsf{OUTG}) \wedge \neg(tuples = \emptyset)\big)$

[] $\langle$ $a, atuple, prType := atuple, \varepsilon, \varepsilon$

$\parallel \langle\ t := \texttt{rdp}(a, \mathbf{T}) \parallel \texttt{retTuple2Usr}(t)\ \rangle$ if $(prName = \mathsf{RDP})$

$\parallel \langle\ \mathbf{t} := \texttt{rdgp}(a, \mathbf{T}) \parallel \texttt{retTuples2Usr}(\mathbf{t})\ \rangle$ if $(prName = \mathsf{RDGP})$

$\parallel \langle\ t := \texttt{inp}(a, \mathbf{T}) \parallel \texttt{retTuple2Usr}(t)\ \rangle$ if $(prName = \mathsf{INP})$

$\parallel \langle\ \mathbf{t} := \texttt{ingp}(a, \mathbf{T}) \parallel \texttt{retTuples2Usr}(\mathbf{t})\ \rangle$ if $(prName = \mathsf{INGP})$

$\rangle$ if $\big((prType = local) \wedge \neg(atuple = \varepsilon)\big)$

{- - - - - - - - - - End of Local Operation Manager - - - - - - - - - -}

{- - - - - - - - - - Start of Remote Operation Manager - - - - - - - - - -}

{Initiate (as reference agent) execution of different remote tuple space operations}

[] $\langle$ $t, tuple, prType := tuple, \varepsilon, \varepsilon \parallel prid := \texttt{getPrID}(prName) \parallel rform := 1$

$\parallel \langle\!\| \ a : 1 \le a \le TAs :: \mathcal{Q}_{T^S_{a_k}} := \mathcal{Q}_{T^S_{a_k}} \bullet \texttt{createRTuple}_{\mathsf{r}}(rform, prid, prName, t, mode, aid, taids[a])\rangle\!\rangle$

$\rangle$ if $\big((prType = remote) \wedge (prName = \mathsf{OUT}) \wedge \neg(tuple = \varepsilon)\big)$

[] $\langle$ $\mathbf{t}, tuples, prType := tuples, \emptyset, \varepsilon \parallel prid := \texttt{getPrID}(prName) \parallel rform := 1$

$\parallel \langle\!\| \ a : 1 \le a \le TAs :: \mathcal{Q}_{T^S_{a_k}} := \mathcal{Q}_{T^S_{a_k}} \bullet \texttt{createRTuple}_{\mathsf{r}}(rform, prid, prName, \mathbf{t}, mode, aid, taids[a])\rangle\!\rangle$

$\rangle$ if $\big((prType = remote) \wedge (prName = \mathsf{OUTG}) \wedge \neg(tuples = \emptyset)\big)$

[] $\langle$ $a, atuple, prType := atuple, \varepsilon, \varepsilon \parallel prid := \texttt{getPrID}(prName) \parallel rform := 1$

$\parallel ROL := ROL \cup \{prid, prName, taids\}$

$\parallel \langle\!\| \ a : 1 \le a \le TAs :: \mathcal{Q}_{T^S_{a_k}} := \mathcal{Q}_{T^S_{a_k}} \bullet \texttt{createRTuple}_{\mathsf{r}}(rform, prid, prName, a, mode, aid, taids[a])\rangle\!\rangle$

$\rangle$ if $\big((prType = remote) \wedge (prRdIn = TRUE) \wedge \neg(atuple = \varepsilon)\big)$

[] $\langle$ $r, \mathcal{Q}_{T^R_{a_k}} := \texttt{head}(\mathcal{Q}_{T^R_{a_k}}), \texttt{tail}(\mathcal{Q}_{T^R_{a_k}})$ $\parallel$ $prid := r \uparrow \text{prid}$

$\parallel \langle\ \mathbb{T}_{prid} := \mathbb{T}_{prid} \cup \{r \uparrow \text{tAid}, r \uparrow \text{data}\} \parallel \langle\ \exists e : (e \in ROL) \wedge (e \uparrow 1 = prid) :: e \uparrow 3 := e \uparrow 3 \setminus r \uparrow \text{tAid}\ \rangle$

$\rangle$ if $\big((r \uparrow \text{rAid} = aid) \wedge \texttt{isPresent}_{\text{in}ROL}(prid, r \uparrow \text{tAid})\big)$ {Handling Response tuple}

$\rangle$ if $\big(\neg(\mathcal{Q}_{T^R_{a_k}} = \perp) \wedge (\texttt{head}(\mathcal{Q}_{T^R_{a_k}}) \uparrow \text{rform} = 2)\big)$

{Return result of execution of remote tuple-reading or -consuming operation to user}

[] $\langle\!\| \ e : (e \in ROL) \wedge (e \uparrow 3 = \emptyset)$

:: $prid, prName := e \uparrow 1, e \uparrow 2 \parallel ROL := ROL \setminus e$

$\parallel \langle\ \langle\!\| \ e : e \in \mathbb{T}_{prid} :: \mathbf{t} := \mathbf{t} \cup e \uparrow \text{tuples}\ \rangle \parallel \texttt{retTuples2Usr}(\mathbf{t})$

$\parallel \langle\!\| \ e : e \in \mathbb{T}_{prid} \wedge \big((prName = \mathsf{ING}) \vee (prName = \mathsf{INGP})\big)$

$:: \mathcal{Q}_{T^S_{a_k}} := \mathcal{Q}_{T^S_{a_k}} \bullet \texttt{createRTuple}_{\mathsf{r}'}(3, prid, prName, aid, e \uparrow \text{tAid})\ \rangle$

$\rangle$ if $\big((prName = \mathsf{RDG}) \vee (prName = \mathsf{RDGP}) \vee (prName = \mathsf{ING}) \vee (prName = \mathsf{INGP})\big)$

$\parallel \langle\ \langle\!\| \ e : e = e'.(e' \in \mathbb{T}_{prid}) :: t, taid := e \uparrow \text{tuple}, e \uparrow \text{tAid}\ \rangle \parallel \texttt{retTuple2Usr}(t)$

$\parallel \mathcal{Q}_{T^S_{a_k}} := \mathcal{Q}_{T^S_{a_k}} \bullet \texttt{createRTuple}_{\mathsf{r}'}(3, prid, prName, aid, taid)$

if $\big((prName = \mathsf{IN}) \vee (prName = \mathsf{INP})\big)$

$\parallel \langle\!\| \ e : e \in \mathbb{T}_{prid} \wedge \neg(e \uparrow \text{tAid} = taid) \wedge \big((prName = \mathsf{IN}) \vee (prName = \mathsf{INP})\big)$

$:: \mathcal{Q}_{T^S_{a_k}} := \mathcal{Q}_{T^S_{a_k}} \bullet \texttt{createRTuple}_{\mathsf{r}'}(4, prid, prName, e \uparrow \text{tuple}, aid, e \uparrow \text{tAid})\ \rangle$

$\rangle$ if $\big((prName = \mathsf{RD}) \vee (prName = \mathsf{RDP}) \vee (prName = \mathsf{IN}) \vee (prName = \mathsf{INP})\big)$

$\rangle$

{- - - - - - - - - - End of Remote Operation Manager - - - - - - - - - -}

Fig. 4. Mobile UNITY **Program** $agent(k)$: part 2

activities of TSMM; its subscripts represent purpose of using it. Also, $\texttt{head}(\mathcal{Q})$ returns front element of $\mathcal{Q}$, while $\texttt{tail}(\mathcal{Q})$ returns all elements of $\mathcal{Q}$ except front element. Again, $\mathcal{Q}\bullet M$ inserts message M in the rear end of $\mathcal{Q}$ and returns updated $\mathcal{Q}$. M comprises of message identity mid, source host's identity src, destination

{- - - - - - - - - - Start of ATS Reaction Manager - - - - - - - - - -}

{Complete execution of different remote tuple space operations}

[] $\langle\ r, \mathcal{Q}_{T^R_{a_k}} := \texttt{head}(\mathcal{Q}_{T^R_{a_k}}), \texttt{tail}(\mathcal{Q}_{T^R_{a_k}})\ \|\ prid := r \uparrow \text{prid}\ \|\ prName := r \uparrow \text{pName}$

$\|\ prBulk := TRUE$

if $\big((prName = \mathsf{RDG}) \vee (prName = \mathsf{RDGP}) \vee (prName = \mathsf{ING}) \vee (prName = \mathsf{INGP})\big)$

$\sim\quad FALSE$

if $\big((prName = \mathsf{RD}) \vee (prName = \mathsf{RDP}) \vee (prName = \mathsf{IN}) \vee (prName = \mathsf{INP})\big)$

$\|\ \langle\ \langle\ t := r \uparrow \text{data} \ \|\ \texttt{out}(t, \mathbf{T})\ \rangle$ if $(prName = \mathsf{OUT})$

$\|\ \langle\ \mathbf{t} := r \uparrow \text{data} \ \|\ \texttt{outg}(\mathbf{t}, \mathbf{T})\ \rangle$ if $(prName = \mathsf{OUTG})$

$\|\ \langle\ a := r \uparrow \text{data} \ \|\ t := \texttt{rd}(a, \mathbf{T})\ \rangle$ if $(prName = \mathsf{RD})$

$\|\ \langle\ a := r \uparrow \text{data} \ \|\ \mathbf{t} := \texttt{rdg}(a, \mathbf{T})\ \rangle$ if $(prName = \mathsf{RDG})$

$\|\ \langle\ a := r \uparrow \text{data} \ \|\ t := \texttt{rdp}(a, \mathbf{T})\ \rangle$ if $(prName = \mathsf{RDP})$

$\|\ \langle\ a := r \uparrow \text{data} \ \|\ \mathbf{t} := \texttt{rdgp}(a, \mathbf{T})\ \rangle$ if $(prName = \mathsf{RDGP})$

$\|\ \langle\ a := r \uparrow \text{data} \ \|\ t := \texttt{in}(a, \mathbf{T})\ \rangle$ if $(prName = \mathsf{IN})$

$\|\ \langle\ a := r \uparrow \text{data} \ \|\ \mathbf{t} := \texttt{ing}(a, \mathbf{T})\ \rangle$ if $(prName = \mathsf{ING})$

$\|\ \langle\ a := r \uparrow \text{data} \ \|\ t := \texttt{inp}(a, \mathbf{T})\ \rangle$ if $(prName = \mathsf{INP})$

$\|\ \langle\ a := r \uparrow \text{data} \ \|\ \mathbf{t} := \texttt{ingp}(a, \mathbf{T})\ \rangle$ if $(prName = \mathsf{INGP})$

$\|\ rform := 2$

$\|\ \mathcal{Q}_{T^S_{a_k}} := \mathcal{Q}_{T^S_{a_k}} \bullet \texttt{createRTuple}_{t'}(rform, prid, prName, t, aid, r \uparrow \text{rAid})$ if $(prBulk = FALSE)$

$\|\ \mathcal{Q}_{T^S_{a_k}} := \mathcal{Q}_{T^S_{a_k}} \bullet \texttt{createRTuple}_{t'}(rform, prid, prName, \mathbf{t}, aid, r \uparrow \text{rAid})$ if $(prBulk = TRUE)$

$\rangle$ if $\big((r \uparrow \text{tAid} = aid) \wedge (r \uparrow \text{rform} = 1)\big)$ {Handling Reaction tuple}

$\|\ \langle\ t := r \uparrow \text{data} \ \|\ \texttt{out}(t, \mathbf{T})$

$\rangle$ if $\big((r \uparrow \text{tAid} = aid) \wedge (r \uparrow \text{rform} = 4)\big)$ {Handling NACK tuple}

$\rangle$ if $\big(\neg(\mathcal{Q}_{T^R_{a_k}} = \perp) \wedge$

$\big((\texttt{head}(\mathcal{Q}_{T^R_{a_k}}) \uparrow \text{rform} = 1) \vee (\texttt{head}(\mathcal{Q}_{T^R_{a_k}}) \uparrow \text{rform} = 3) \vee (\texttt{head}(\mathcal{Q}_{T^R_{a_k}}) \uparrow \text{rform} = 4)\big)\big)$

{- - - - - - - - - - End of ATS Reaction Manager - - - - - - - - - -}

{Discard messages destined for other agents}

[] $\mathcal{Q}_{T^R_{a_k}} := \texttt{tail}(\mathcal{Q}_{T^R_{a_k}})$ if $\big(\neg(\mathcal{Q}_{T^R_{a_k}} = \perp) \wedge \neg(\texttt{head}(\mathcal{Q}_{T^R_{a_k}}) \uparrow \text{dstAg} = aid)\big)$

end

Fig. 5. Mobile UNITY **Program** $agent(k)$: part 3

host's identity *dest*, type of message *kind*, data encapsulated within it *data*, and network interface, *ni*, through which M will be transmitted. M is generated by calling `newMsg`$(src, dest, kind, data, ni)$, which inserts its *mid* to return a complete message. Possible types of messages included in these specifications are BCON, RT, ACK, Locate, and Found messages.

3.1 Formalization of *agent*(k)

Each agent is represented by program $agent(k)$, which comprises of ***declare***, ***always***, ***initially*** and ***assign*** sections. Agent behavior is specified by different variables that are declared in ***declare***, like *aid* and *type* as agent identity and nature of $agent(k)$. **T** is declared as `ATS` of $agent(k)$, and *prid* as identity of invoked primitive of $agent(k)$. *ROL* is declared as remote operation list of $agent(k)$, and *RL* is declared as reactive list of $agent(k)$. $\mathcal{Q}_{T^S_{a_k}}$ and $\mathcal{Q}_{T^R_{a_k}}$ are declared as queues to interface between agents and their supported hosts, and transfer request/response

Program $host(i)$ **at** λ

declare

$type$: $\in\{stationary, mobile, accesspoint\}$
[] hid : hostid [] $assoc$: set of hostid
[] $nwdeploy$: $\in\{iBSS, IBSS\}$ [] $status$: $\in\{standalone, connected, associated\}$
[] $\mathbf{T}'$: tuple space
[] $\mathcal{Q}_{T^S_{a_k}}, \mathcal{Q}_{T^R_{a_k}}$: queue of RT_{tuple} [] $\mathcal{Q}_{RT_S}, \mathcal{Q}_{RT_R}$: queue of RT_{tuple} [] r : RT_{tuple}
[] a : agentid [] $\mathcal{A}$: set of agentid [] $\mathcal{Q}_{in}, \mathcal{Q}_{out}$: queue of agentid
[] $\mathcal{H}$: set of ($\text{MH}_{\text{hostid}}$, $\text{AP}_{\text{hostid}}$, timestamp) [] $\mathcal{L}$: set of ($\text{MH}_{\text{hostid}}$, RT_{tuple}, timestamp)
[] $\mathcal{CS}$: message [] $\mathcal{T}_{\text{W}}, \mathcal{T}_{\text{WL}}$: message [] M, m : message
[] $\mathcal{L}RT$: set of ($\text{AP}_{\text{hostid}}/\text{MH}_{\text{hostid}}$, RT_{msgid})
[] $\mathcal{N}$: set of ($\text{Host}_{\text{hostid}}$, set of agentid, timestamp, extant)
[] $\mathcal{Q}_{S_B}, \mathcal{Q}_{R_B}$: queue of message [] $\mathcal{Q}_{S_{RT}}, \mathcal{Q}_{R_{RT}}$: queue of message
[] $\mathcal{Q}_{S_{\text{W}}}, \mathcal{Q}_{S_{\text{WL}}}$: queue of message [] $\mathcal{Q}_S, \mathcal{Q}_R$: queue of message
[] $clock, lastHTSchk, lastRTsent, lastBsent, newRTGap, rtAtmpt$: natural

always

$B_{iBSS_{\text{W}}}$ = **IBSSBroadcastAddress$_{\text{DS}}$** [] $B_{iBSS_{\text{WL}}}$ = **IBSSBroadcastAddress$_{\text{BSA}}$**
[] $B_{IBSS_{\text{WL}}}$ = **IBSSBroadcastAddress**
[] λ := `Location`(i) [] hid := `getMyHostID`(i)
[] $type$:= `getHostType`($stationary, mobile, accesspoint$)
[] $nwdeploy$:= `getNetwork`($iBSS, IBSS$)
[] $mhGap$ = SystemMHValidityInterval [] $HTSaccessGap$ = SystemHTSAccessInterval
[] $locateGap$ = SystemLocateMsgInterval [] $bconGap$ = SystemBeaconInterval
[] $mhGap$ = SystemMHValidityInterval [] $bLife$ = SystemBeaconLifetime
[] $\texttt{isPresent}_{\mathcal{H}}(mhid) \equiv \langle\ \exists e : (e \in \mathcal{H}) \wedge (e \uparrow 1 = mhid)\ \rangle$
[] $\texttt{isPresent}_{\mathcal{L}}(mhid) \equiv \langle\ \exists e : (e \in \mathcal{L}) \wedge (e \uparrow 1 = mhid)\ \rangle$
[] $\texttt{isPresent}_{\mathcal{N}}(hostid) \equiv \langle\ \exists e : (e \in \mathcal{N}) \wedge (e \uparrow 1 = hostid)\ \rangle$
[] $\texttt{isPresent}_{\mathcal{L}RT}(hostid) \equiv \langle\ \exists e : (e \in \mathcal{L}RT) \wedge (e \uparrow 1 = hostid)\ \rangle$
[] $\texttt{isRepeat}_{\mathcal{L}RT}(hostid, msgid) \equiv \langle\ \exists e : (e \in \mathcal{L}RT) \wedge (e \uparrow 1 = hostid) \wedge (e \uparrow 2 = msgid)\ \rangle$
[] $\texttt{isValid}_{\mathcal{H}}(e, now) \equiv \big((e \in \mathcal{H}) \wedge ((now - e \uparrow 3) \leq mhGap)\big)$
[] $\texttt{isValid}_{\mathcal{L}}(e, now) \equiv \big((e \in \mathcal{L}) \wedge \big((now - e \uparrow 3) \leq locateGap\big)\big)$
[] $\texttt{isValid}_{\mathcal{N}}(e, now) \equiv \big((e \in \mathcal{N}) \wedge \big((now - e \uparrow 3) \leq e \uparrow 4\big)\big)$
[] $\texttt{isMsgBcon}(msg) \equiv (msg{\cdot}\, kind = Beacon)$
[] $\texttt{isMsgRT}(msg) \equiv (msg{\cdot}\, kind = RT)$ [] $\texttt{isMsgACK}(msg) \equiv (msg{\cdot}\, kind = ACK)$
[] $\texttt{isMsgLocate}(msg) \equiv (msg{\cdot}\, kind = Locate)$ [] $\texttt{isMsgFound}(msg) \equiv (msg{\cdot}\, kind = Found)$
[] $\texttt{isNotOwnMsg}(msg) \equiv \neg(msg{\cdot}\, src = hid)$
[] $\texttt{isSH}(host) \equiv (host{\cdot}\, type = stationary)$ [] $\texttt{isMH}(host) \equiv (host{\cdot}\, type = mobile)$
[] $\texttt{isAP}(host) \equiv (host{\cdot}\, type = accesspoint)$

Fig. 6. Mobile UNITY **Program** $host(i)$: part 1

tuples from agents to hosts and vice versa. When user application is invoking any tuple space operation, corresponding agent captures different parameters required to complete that operation.

3.2 Formalization of *host*(i)

Like $agent(k)$, $host(i)$ also comprises of ***declare***, ***always***, ***initially*** and ***assign*** sections. Different variables related to host behavior is declared in ***declare*** section,

initially

$clock = 0$ ▯ $lastHTSchk = 0$ ▯ $lastRTsent = 0$ ▯ $lastBsent = 0$
▯ $status = standalone$ ▯ $assoc = \emptyset$ ▯ $\mathcal{H} = \emptyset$ ▯ $\mathcal{L} = \emptyset$ ▯ $\mathcal{LRT} = \emptyset$ ▯ $\mathcal{A} = \emptyset$ ▯ $\mathcal{N} = \emptyset$
▯ $\mathbf{T}' = \perp$ ▯ $\mathcal{T}_{\mathrm{W}} = \perp$ ▯ $\mathcal{T}_{\mathrm{WL}} = \perp$ ▯ $\mathcal{CS} = \perp$
▯ $\mathcal{Q}_{T^S_{a_k}} = \perp$ ▯ $\mathcal{Q}_{T^R_{a_k}} = \perp$ ▯ $\mathcal{Q}_{in} = \perp$ ▯ $\mathcal{Q}_{out} = \perp$ ▯ $\mathcal{Q}_{RT_S} = \perp$ ▯ $\mathcal{Q}_{RT_R} = \perp$
▯ $\mathcal{Q}_{S_B} = \perp$ ▯ $\mathcal{Q}_{R_B} = \perp$ ▯ $\mathcal{Q}_{S_{RT}} = \perp$ ▯ $\mathcal{Q}_{R_{RT}} = \perp$ ▯ $\mathcal{Q}_{S_{\mathrm{W}}} = \perp$ ▯ $\mathcal{Q}_{S_{\mathrm{WL}}} = \perp$ ▯ $\mathcal{Q}_S = \perp$ ▯ $\mathcal{Q}_R = \perp$

assign

{Increment the clock}
▯ $clock := clock + 1$

{- - - - - - - - - - Start of Transport Interface - - - - - - - - - -}

{Organize a message for onward transmission}
▯ ⟨ $M, \mathcal{Q}_S := \mathtt{head}(\mathcal{Q}_S), \mathtt{tail}(\mathcal{Q}_S)$
‖ ⟨ $\mathcal{Q}_{S_{\mathrm{W}}} := \mathcal{Q}_{S_{\mathrm{W}}} \bullet M$ if $(M \cdot ni = \mathrm{W})$ ‖ $\mathcal{Q}_{S_{\mathrm{WL}}} := \mathcal{Q}_{S_{\mathrm{WL}}} \bullet M$ if $(M \cdot ni = \mathrm{WL})$ ⟩
⟩ if $\neg(\mathcal{Q}_S = \perp)$

{Transfer a message from $\mathcal{Q}_{S_{\mathrm{W}}}$ to $\mathcal{T}_{\mathrm{W}}$; make $\mathcal{T}_{\mathrm{W}}$ empty after some time}
▯ $transmit\&reset_{\mathrm{W}}$ :: ⟨ $\mathcal{T}_{\mathrm{W}}, \mathcal{Q}_{S_{\mathrm{W}}} := \mathtt{head}(\mathcal{Q}_{S_{\mathrm{W}}}), \mathtt{tail}(\mathcal{Q}_{S_{\mathrm{W}}})$ if $\neg(\mathcal{Q}_{S_{\mathrm{W}}} = \perp) \wedge (\mathcal{T}_{\mathrm{W}} = \perp)$;
$\mathcal{T}_{\mathrm{W}} := \perp$ ⟩

{Transfer a message from $\mathcal{Q}_{S_{\mathrm{WL}}}$ to $\mathcal{T}_{\mathrm{WL}}$; make $\mathcal{T}_{\mathrm{WL}}$ empty after some time}
▯ $transmit\&reset_{\mathrm{WL}}$ :: ⟨ $\mathcal{T}_{\mathrm{WL}}, \mathcal{Q}_{S_{\mathrm{WL}}} := \mathtt{head}(\mathcal{Q}_{S_{\mathrm{WL}}}), \mathtt{tail}(\mathcal{Q}_{S_{\mathrm{WL}}})$ if $\neg(\mathcal{Q}_{S_{\mathrm{WL}}} = \perp) \wedge (\mathcal{T}_{\mathrm{WL}} = \perp)$;
$\mathcal{T}_{\mathrm{WL}} := \perp$ ⟩

{Transfer a message from $\mathcal{T}_{\mathrm{W}}$ to $\mathcal{Q}_R$}
▯ ⟨ $\mathcal{Q}_R := \mathcal{Q}_R \bullet \mathcal{T}_{\mathrm{W}}$ if $\mathtt{isNotOwnMsg}(\mathcal{T}_{\mathrm{W}})$ ⟩ reacts-to $\neg(\mathcal{T}_{\mathrm{W}} = \perp)$

{Transfer a message from $\mathcal{T}_{\mathrm{WL}}$ to $\mathcal{Q}_R$}
▯ ⟨ $\mathcal{Q}_R := \mathcal{Q}_R \bullet \mathcal{T}_{\mathrm{WL}}$ if $\mathtt{isNotOwnMsg}(\mathcal{T}_{\mathrm{WL}})$ ⟩ reacts-to $\neg(\mathcal{T}_{\mathrm{WL}} = \perp)$

{Organize a received Beacon/RT/ACK/Locate/Found message for further processing}
▯ ⟨ $M, \mathcal{Q}_R := \mathtt{head}(\mathcal{Q}_R), \mathtt{tail}(\mathcal{Q}_R)$
‖ ⟨ $\mathcal{Q}_{R_B} := \mathcal{Q}_{R_B} \bullet M$ if $\mathtt{isMsgBcon}(M)$
‖ $\mathcal{Q}_{R_{RT}} := \mathcal{Q}_{R_{RT}} \bullet M$ if $\mathtt{isMsgRT}(M) \vee \mathtt{isMsgACK}(M) \vee \mathtt{isMsgLocate}(M) \vee \mathtt{isMsgFound}(M)$
⟩
⟩ if $\neg(\mathcal{Q}_R = \perp)$

{- - - - - - - - - - End of Transport Interface - - - - - - - - - -}

Fig. 7. Mobile UNITY **Program** $host(i)$: part 2

like *hid* as host identity of $host(i)$ and *type* as nature of $host(i)$. $\mathbf{T}'$ is declared as its `HTS`. $\mathcal{H}$ and $\mathcal{L}$ are declared for `History` (that records path of successful data transfer to different mobile hosts) and location list (that lists mobile hosts with ongoing location search) respectively for $host(i)$ of stationary hosts and access points in iBSS. Moreover, $\mathcal{LRT}$ and $\mathcal{CS}$ are declared for `LastRT` (that records message identity of last data messages received from different hosts) and `CommStash` (that buffers data messages) respectively of $host(i)$ of mobile hosts and access points in both iBSS and IBSS. Also, $\mathcal{N}$ and $\mathcal{A}$ are declared to represent `NeighborList` and `AgentList` respectively. Different macros related to various aspects of discovery and communication mechanisms are included in appendix.

At lowest level, TSMM interacts with transport service, which is formalized as Transport Interface by a set of assignment statements. Discovery Manager and Communication Manager interchange messages with Transport Interface through $\mathcal{Q}_S$

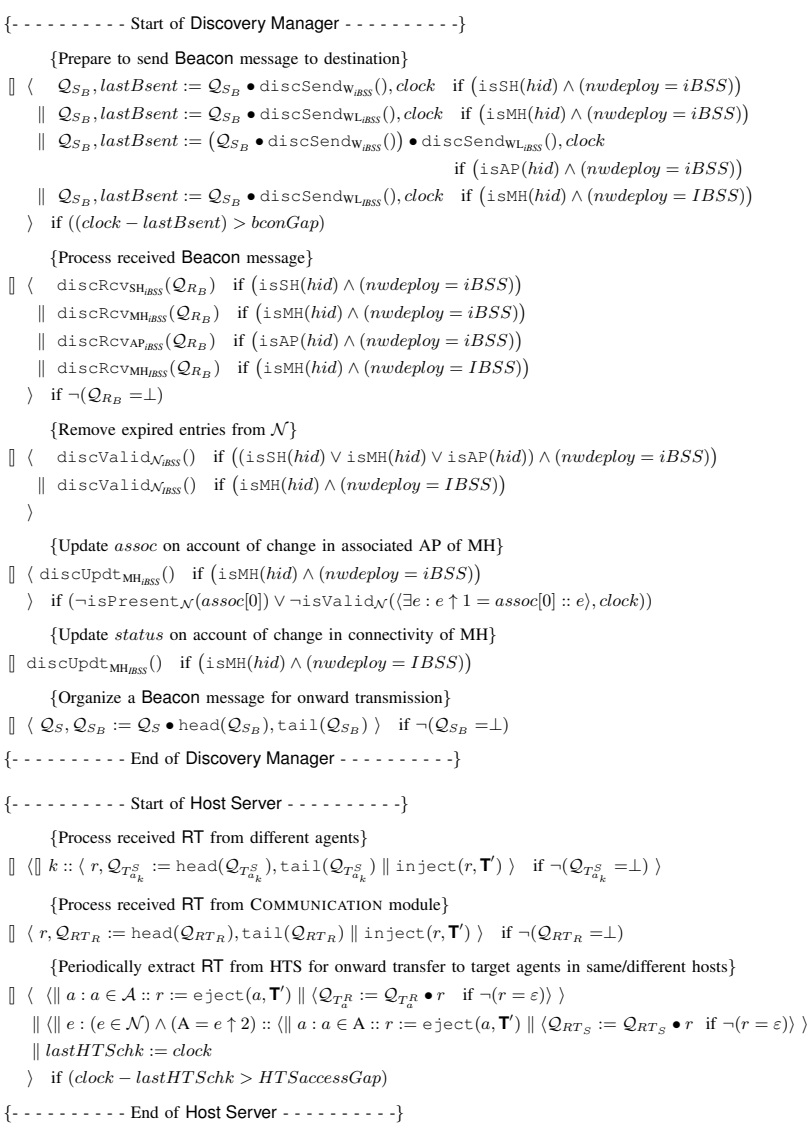

Fig. 8. Mobile UNITY **Program** $host(i)$: part 3

and $\mathcal{Q}_R$. Some behaviors of Discovery Manager and Communication Manager are abstracted as macros, which are used in different assignment statements to fulfill all functionalities of Discovery Manager and Communication Manager. Host Server interchanges request/response tuples (represented as $\mathrm{RT}_{\mathrm{tuple}}$) with Communication Manager through $\mathcal{Q}_{RT_S}$ and $\mathcal{Q}_{RT_R}$, which is formalized via a set of assignment statements. Similarly, a pair of assignment statements formalizes registration/deregistration functionalities of Agent Manager.

{- - - - - - - - - - Start of Communication Manager - - - - - - - - - -}

{Prepare to send RT/Locate message to destination}

$$\begin{array}{l}[\!] \ \langle \quad \mathtt{commSend}_{\mathtt{SH}_{iBSS}}(\mathcal{Q}_{RT_S}) \quad \text{if } \big(\mathtt{isSH}(hid) \wedge (nwdeploy = iBSS)\big) \\ \quad \| \ \mathtt{commSend}_{\mathtt{MH}_{iBSS}}(\mathcal{Q}_{RT_S}) \quad \text{if } \big(\mathtt{isMH}(hid) \wedge (nwdeploy = iBSS)\big) \\ \quad \| \ \mathtt{commSend}_{\mathtt{AP}_{iBSS}}(\mathcal{Q}_{RT_S}) \quad \text{if } \big(\mathtt{isAP}(hid) \wedge (nwdeploy = iBSS)\big) \\ \quad \| \ \mathtt{commSend}_{\mathtt{MH}_{IBSS}}(\mathcal{Q}_{RT_S}) \quad \text{if } \big(\mathtt{isMH}(hid) \wedge (nwdeploy = IBSS)\big) \\ \ \rangle \quad \text{if } \neg(\mathcal{Q}_{RT_S} = \bot)\end{array}$$

{Process received RT/Locate/Found message, and prepare to send RT/ACK/Found message}

$$\begin{array}{l}[\!] \ \langle \quad \mathtt{commRcv}_{\mathtt{SH}_{iBSS}}(\mathcal{Q}_{R_{RT}}) \quad \text{if } \big(\mathtt{isSH}(hid) \wedge (nwdeploy = iBSS)\big) \\ \quad \| \ \mathtt{commRcv}_{\mathtt{MH}_{iBSS}}(\mathcal{Q}_{R_{RT}}) \quad \text{if } \big(\mathtt{isMH}(hid) \wedge (nwdeploy = iBSS)\big) \\ \quad \| \ \mathtt{commRcv}_{\mathtt{AP}_{iBSS}}(\mathcal{Q}_{R_{RT}}) \quad \text{if } \big(\mathtt{isAP}(hid) \wedge (nwdeploy = iBSS)\big) \\ \quad \| \ \mathtt{commRcv}_{\mathtt{MH}_{IBSS}}(\mathcal{Q}_{R_{RT}}) \quad \text{if } \big(\mathtt{isMH}(hid) \wedge (nwdeploy = IBSS)\big) \\ \ \rangle \quad \text{if } \neg(\mathcal{Q}_{R_{RT}} = \bot)\end{array}$$

{Resend RT message whose ACK fails to reach before timeout}

$$\begin{array}{l}[\!] \ \langle \quad \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \mathtt{commReSend}_{\mathtt{RT}_{iBSS}}() \quad \text{if } \big((\mathtt{isMH}(hid) \vee \mathtt{isAP}(hid)) \wedge (nwdeploy = iBSS)\big) \\ \quad \| \ \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \mathtt{commReSend}_{\mathtt{RT}_{IBSS}}() \quad \text{if } \big(\mathtt{isMH}(hid) \wedge (nwdeploy = IBSS)\big) \\ \ \rangle \quad \text{if } ((clock - lastRTsent) > newRTGap)\end{array}$$

{Process RT message whose destination is presently not available}

$$\begin{array}{l}[\!] \ \langle \quad \langle \ \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet \mathcal{CS}.data \ \| \ \mathcal{CS} := \bot \ \rangle \quad \text{if } \big(\mathtt{isMH}(hid) \wedge (nwdeploy = iBSS)\big) \\ \quad \| \ \langle \quad \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \mathtt{newMsg}\big(hid, B_{iBSS_W}, Locate, \mathcal{CS}.dest, \mathrm{W}\big) \\ \qquad \| \ \mathcal{L} := \mathcal{L} \cup \{(\mathcal{CS}.dest, \mathcal{CS}.data, clock)\} \ \rangle \quad \text{if } \big(\mathtt{isAP}(hid) \wedge (nwdeploy = iBSS)\big) \\ \quad \| \ \langle \ \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet \mathcal{CS}.data \ \| \ \mathcal{CS} := \bot \ \rangle \quad \text{if } \big(\mathtt{isMH}(hid) \wedge (nwdeploy = IBSS)\big) \\ \ \rangle \quad \text{if } \big(\neg(\mathcal{CS} = \bot) \wedge (rtAtmpt > 3)\big)\end{array}$$

{Remove expired entries from $\mathcal{H}$ and $\mathcal{L}$, and preserve unsent RT}

$$[\!] \ \mathtt{commValid}_{\mathcal{H}\mathcal{L}_{iBSS}}() \quad \text{if } \big((\mathtt{isSH}(hid) \vee \mathtt{isAP}(hid)) \wedge (nwdeploy = iBSS)\big)$$

{Organize RT/ACK/Locate/Found message for onward transmission}

$$[\!] \ \langle \ \mathcal{Q}_S, \mathcal{Q}_{S_{RT}} := \mathcal{Q}_S \bullet \mathtt{head}(\mathcal{Q}_{S_{RT}}), \mathtt{tail}(\mathcal{Q}_{S_{RT}}) \ \rangle \quad \text{if } \neg(\mathcal{Q}_{S_{RT}} = \bot)$$

{- - - - - - - - - - End of Communication Manager - - - - - - - - - -}

{- - - - - - - - - - Start of Agent Manager - - - - - - - - - -}

{Register active agents in $\mathcal{A}$}

$$[\!] \ \mathcal{A}, \mathcal{Q}_{in} := \mathcal{A} \cup \mathtt{head}(\mathcal{Q}_{in}), \mathtt{tail}(\mathcal{Q}_{in}) \quad \text{if } \neg(\mathcal{Q}_{in} = \bot)$$

{Deregister terminated/migrated agents from $\mathcal{A}$}

$$[\!] \ \mathcal{A}, \mathcal{Q}_{out} := \mathcal{A} \setminus \mathtt{head}(\mathcal{Q}_{out}), \mathtt{tail}(\mathcal{Q}_{out}) \quad \text{if } \big(\neg(\mathcal{Q}_{out} = \bot) \wedge (\mathtt{head}(\mathcal{Q}_{out}) \in \mathcal{A})\big)$$

{- - - - - - - - - - End of Agent Manager - - - - - - - - - -}

end

Fig. 9. Mobile UNITY **Program** $host(i)$: part 4

4 Conclusion

This paper has proposed an approach of formalization of a TSMM, which decouples coordination among interacting agents of supported applications when deployed over multiple heterogeneous mobile, dynamic and unreliable networks. Proposed approach formally specifies TSMM as a Mobile UNITY system, comprising of components representing different behaviors of agents and hosts of TSMM. This formalization can be shown to reason TSMM as an appropriate coordination platform for multiple underlying heterogeneous networks, which facilitates development of robust and flexible mobile computing applications.

References

1. Bruneo, D., Puliafito, A., Scarpa, M.: Mobile Middleware: Definition and Motivations. In: Bellavista, P., Corradi, A. (eds.) The Handbook of Mobile Middleware, pp. 145–167. Auerbach Pub. (2007)
2. Gelernter, D.: Generative Communication in Linda. Transactions on Programming Languages and Systems 7(1), 80–112 (1985)
3. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of Publish/Subscribe. Computing Surveys 35(2), 114–131 (2003)
4. De, S., Nandi, S., Goswami, D.: Architectures of Mobile Middleware: A Taxonomic Perspective. In: Proc. 2nd IEEE Intl. Conf. on Parallel, Distributed and Grid Computing, PDGC 2012 (December 2012)
5. De, S., Nandi, S., Goswami, D.: Modeling an Enhanced Tuple Space based Mobile Middleware in UNITY. In: Proc. 11th IEEE Intl. Conf. on Ubiquitous Computing and Communications, IUCC 2012, pp. 1684–1691 (June 2012)
6. De, S., Goswami, D., Nandi, S., Chakraborty, S.: Formalization of a Fully-Decoupled Reactive Tuple Space Model for Mobile Middleware. In: Borcea, C., Bellavista, P., Gianelli, C., Magedanz, T., Schreiner, F. (eds.) Mobilware 2012. LNICST, vol. 65, pp. 77–91. Springer, Heidelberg (2013)
7. Roman, G.C., McCann, P.J., Plun, J.Y.: Mobile UNITY: Reasoning and Specification in Mobile Computing. Transactions on Software Engineering and Methodology 6(3), 250–282 (1997)
8. Murphy, A.L., Picco, G.P., Roman, G.C.: Lime: A Coordination Model and Middleware supporting Mobility of Hosts and Agents. Transactions on Software Engineering and Methodology 15(3), 279–328 (2006)
9. Roman, G.C., Payton, J.: Mobile UNITY Schemas for Agent Coordination. In: Börger, E., Gargantini, A., Riccobene, E. (eds.) ASM 2003. LNCS, vol. 2589, pp. 126–150. Springer, Heidelberg (2003)
10. IEEE 802.11 WG Std.: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical Report 802.11$^{\mathrm{TM}}$ (June 2007)
11. De, S., Nandi, S., Goswami, D.: On Performance Improvement Issues in Unordered Tuple Space based Mobile Middleware. In: Proc. 2010 Annual IEEE India Conf., INDICON 2010 (December 2010)
12. De, S., Goswami, D., Nandi, S.: A New Tuple Space Structure for Tuple Space based Mobile Middleware Platforms. In: Proc. 2012 Annual IEEE India Conf., INDICON 2012 (December 2012)
13. De, S., Chakraborty, S., Nandi, S., Goswami, D.: Supporting Tuple Space based Mobile Middleware over Unreliable Mobile Infrastructures: Design and Formal Specifications. In: Proc. 6th IEEE Intl. Conf. on Advanced Networks and Telecommunications Systems, ANTS 2012 (December 2012)

A Appendix: Macros Related to Formalization of TSMM

A.1 Macros of Discovery Manager

$M := \texttt{discSend}_{\mathrm{W}_{iBSS}}() \triangleq \langle\ M := \texttt{newMsg}(hid, B_{iBSS_{\mathrm{W}}}, BCON, \texttt{buildBcon}(\mathcal{A}, bLife), \mathrm{W})\ \rangle$

$M := \texttt{discSend}_{\mathrm{WL}_{iBSS}}() \triangleq \langle\ M := \texttt{newMsg}(hid, B_{iBSS_{\mathrm{WL}}}, BCON, \texttt{buildBcon}(\mathcal{A}, bLife), \mathrm{WL})\ \rangle$

$M := \texttt{discSend}_{\mathrm{WL}_{IBSS}}() \triangleq \langle\ M := \texttt{newMsg}(hid, B_{IBSS_{\mathrm{WL}}}, BCON, \texttt{buildBcon}(\mathcal{A}, bLife), \mathrm{WL})\ \rangle$

$$
\begin{array}{l}
\texttt{discRcv}_{\mathrm{SH}_{iBSS}}(\mathcal{Q}_{R_B}) \triangleq \\
\quad \langle\ m, \mathcal{Q}_{R_B} := \texttt{head}(\mathcal{Q}_{R_B}), \texttt{tail}(\mathcal{Q}_{R_B}) \\
\quad \| \langle\ \mathcal{N} := \mathcal{N} \cup \{m.src, m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant}\} \quad \text{if } \neg\texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad \| \langle \exists e : (e \in \mathcal{N}) \wedge (e \uparrow 1 = m.src) :: e \uparrow 2, e \uparrow 3, e \uparrow 4 := m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant} \\
\qquad\quad \rangle \quad \text{if } \texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad \rangle \quad \text{if } (\texttt{isSH}(m.src) \vee \texttt{isAP}(m.src)) \\
\quad \rangle \\
\texttt{discRcv}_{\mathrm{MH}_{iBSS}}(\mathcal{Q}_{R_B}) \triangleq \\
\quad \langle\ m, \mathcal{Q}_{R_B} := \texttt{head}(\mathcal{Q}_{R_B}), \texttt{tail}(\mathcal{Q}_{R_B}) \\
\quad \| \langle\ \mathcal{N} := \mathcal{N} \cup \{m.src, m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant}\} \quad \text{if } \neg\texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad \| \langle \exists e : (e \in \mathcal{N}) \wedge (e \uparrow 1 = m.src) :: e \uparrow 2, e \uparrow 3, e \uparrow 4 := m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant} \\
\qquad\quad \rangle \quad \text{if } \texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad \| status, assoc[0] := associated, m.src \\
\qquad \rangle \quad \text{if } (\texttt{isAP}(m.src) \wedge (status = standalone)) \\
\quad \| \langle\ \mathcal{N} := \mathcal{N} \cup \{m.src, m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant}\} \quad \text{if } \neg\texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad \| \langle \exists e : (e \in \mathcal{N}) \wedge (e \uparrow 1 = m.src) :: e \uparrow 2, e \uparrow 3, e \uparrow 4 := m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant} \\
\qquad\quad \rangle \quad \text{if } \texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad \| assoc[0] := m.src \\
\qquad \rangle \quad \text{if } (\texttt{isAP}(m.src) \wedge (status = associated)) \\
\quad \rangle \\
\texttt{discRcv}_{\mathrm{AP}_{iBSS}}(\mathcal{Q}_{R_B}) \triangleq \\
\quad \langle\ \ m, \mathcal{Q}_{R_B} := \texttt{head}(\mathcal{Q}_{R_B}), \texttt{tail}(\mathcal{Q}_{R_B}) \\
\quad\ \| \langle\ \ \mathcal{N} := \mathcal{N} \cup \{m.src, m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant}\} \quad \text{if } \neg\texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad \| \langle \exists e : (e \in \mathcal{N}) \wedge (e \uparrow 1 = m.src) :: e \uparrow 2, e \uparrow 3, e \uparrow 4 := m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant} \\
\qquad\quad \rangle \quad \text{if } \texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad \rangle \quad \text{if } (\texttt{isSH}(m.src) \vee \texttt{isAP}(m.src)) \\
\quad\ \| \langle\ \ \mathcal{N} := \mathcal{N} \cup \{m.src, m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant}\} \quad \text{if } \neg\texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad \| \langle \exists e : (e \in \mathcal{N}) \wedge (e \uparrow 1 = m.src) :: e \uparrow 2, e \uparrow 3, e \uparrow 4 := m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant} \\
\qquad\quad \rangle \quad \text{if } \texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad \| assoc := assoc \cup \{m.src\} \\
\qquad \rangle \quad \text{if } \texttt{isMH}(m.src) \\
\quad \rangle \\
\texttt{discUpdt}_{\mathrm{MH}_{iBSS}}() \triangleq \langle\ \ assoc[0] := \perp \ \ \| \ \ \langle assoc[0] := a.(a \in \mathcal{N}) \quad \text{if } \neg(\mathcal{N} = \emptyset) \rangle \\
\qquad\qquad\qquad\qquad \| \ status := standalone \quad \text{if } (assoc[0] = \perp) \qquad \sim \qquad associated \quad \text{if } \neg(assoc[0] = \perp) \\
\qquad\qquad\qquad\qquad \rangle \\
\texttt{discValid}_{\mathcal{N}_{iBSS}}() \triangleq \langle \| \ e : e \in \mathcal{N} :: \mathcal{N} := \mathcal{N} \setminus \{e\} \quad \text{if } \neg\texttt{isValid}_{\mathcal{N}}(e, clock) \ \rangle \\
\texttt{discUpdt}_{\mathrm{MH}_{IBSS}}() \triangleq \langle\ \ status := standalone \quad \text{if } (\mathcal{N} = \emptyset) \qquad \sim \qquad connected \quad \text{if } \neg(\mathcal{N} = \emptyset) \quad \rangle \\
\texttt{discValid}_{\mathcal{N}_{IBSS}}() \triangleq \langle \| \ e : e \in \mathcal{N} :: \mathcal{N} := \mathcal{N} \setminus \{e\} \quad \text{if } \neg\texttt{isValid}_{\mathcal{N}}(e, clock) \ \rangle
\end{array}
\tag{1}
$$

$$
\begin{array}{l}
\texttt{discRcv}_{\mathrm{MH}_{IBSS}}(\mathcal{Q}_{R_B}) \triangleq \\
\quad \langle\ \ m, \mathcal{Q}_{R_B} := \texttt{head}(\mathcal{Q}_{R_B}), \texttt{tail}(\mathcal{Q}_{R_B}) \\
\quad\ \| \langle\ \ \mathcal{N} := \mathcal{N} \cup \{m.src, m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant}\} \quad \text{if } \neg\texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad\quad \| \langle \exists e : (e \in \mathcal{N}) \wedge (e \uparrow 1 = m.src) :: e \uparrow 2, e \uparrow 3, e \uparrow 4 := m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant} \\
\qquad\quad\ \rangle \quad \text{if } \texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad\quad \| \ status := connected \\
\qquad\quad \rangle \quad \text{if } (\texttt{isMH}(m.src) \wedge (status = standalone)) \\
\quad\ \| \langle\ \ \mathcal{N} := \mathcal{N} \cup \{m.src, m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant}\} \quad \text{if } \neg\texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad\quad \| \langle \exists e : (e \in \mathcal{N}) \wedge (e \uparrow 1 = m.src) :: e \uparrow 2, e \uparrow 3, e \uparrow 4 := m.data \uparrow \text{agids}, clock, m.data \uparrow \text{extant} \\
\qquad\quad\ \rangle \quad \text{if } \texttt{isPresent}_{\mathcal{N}}(m.src) \\
\qquad\quad \rangle \quad \text{if } (\texttt{isMH}(m.src) \wedge (status = connected)) \\
\quad\ \rangle
\end{array}
$$

A.2 Macros of Communication Manager

$$\texttt{commSend}_{\mathrm{SH}_{iBSS}}(\mathcal{Q}_{RT_S}) \triangleq$$

$$\langle \quad r, \mathcal{Q}_{RT_S} := \texttt{head}(\mathcal{Q}_{RT_S}), \texttt{tail}(\mathcal{Q}_{RT_S}) \quad \| \quad dstid := r \uparrow \mathrm{dstHost}$$

$$\| \quad \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, dstid, RT, r, \mathrm{W}) \quad \text{if } (\texttt{isSH}(dstid) \wedge (host_{hid} \Gamma' host_{dstid}))$$

$$\| \quad \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet r \quad \text{if } (\texttt{isSH}(dstid) \wedge \neg(host_{hid} \Gamma' host_{dstid}))$$

$$\| \quad \langle \quad apid := \langle\ \exists\ e : (e \in \mathcal{H}) \wedge (e \uparrow 1 = dstid) :: e \uparrow 2\ \rangle$$

$$\| \quad \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, apid, RT, r, \mathrm{W}) \quad \text{if } (host_{hid} \Gamma' host_{apid})$$

$$\| \quad \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet r \quad \text{if } \neg(host_{hid} \Gamma' host_{apid})$$

$$\rangle \quad \text{if } (\texttt{isMH}(dstid) \wedge \texttt{isPresent}_{\mathcal{H}}(dstid))$$

$$\| \quad \langle \ \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, B_{iBSS_{\mathrm{W}}}, Locate, dstid, \mathrm{W}) \ \| \ \mathcal{L} := \mathcal{L} \cup \{(dstid, r, clock)\}$$

$$\rangle \quad \text{if } (\texttt{isMH}(dstid) \wedge \neg\texttt{isPresent}_{\mathcal{H}}(dstid))$$

$$\rangle$$

$$\texttt{commSend}_{\mathrm{MH}_{iBSS}}(\mathcal{Q}_{RT_S}) \triangleq$$

$$\langle \quad r, \mathcal{Q}_{RT_S} := \texttt{head}(\mathcal{Q}_{RT_S}), \texttt{tail}(\mathcal{Q}_{RT_S}) \quad \| \quad m := \texttt{newMsg}(hid, assoc[0], RT, r, \mathrm{WL})$$

$$\| \quad \mathcal{Q}_{S_{RT}}, \mathcal{CS}, lastRTsent, newRTGap, rtAtmpt := \mathcal{Q}_{S_{RT}} \bullet m, m, clock, rtGap, 0$$

$$\text{if } (host_{hid} \Gamma' host_{assoc[0]})$$

$$\| \quad \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet r \quad \text{if } \neg(host_{hid} \Gamma' host_{assoc[0]})$$

$$\rangle \quad \text{if } ((status = associated) \wedge (\mathcal{CS} = \perp))$$

$$\texttt{commSend}_{\mathrm{AP}_{iBSS}}(\mathcal{Q}_{RT_S}) \triangleq$$

$$\langle \quad r, \mathcal{Q}_{RT_S} := \texttt{head}(\mathcal{Q}_{RT_S}), \texttt{tail}(\mathcal{Q}_{RT_S}) \quad \| \quad dstid := r \uparrow \mathrm{dstHost}$$

$$\| \quad \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, dstid, RT, r, \mathrm{W}) \quad \text{if } (\texttt{isSH}(dstid) \wedge (host_{hid} \Gamma' host_{dstid}))$$

$$\| \quad \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet r \quad \text{if } (\texttt{isSH}(dstid) \wedge \neg(host_{hid} \Gamma' host_{dstid}))$$

$$\| \quad \langle \quad m := \texttt{newMsg}(hid, dstid, RT, r, \mathrm{WL})$$

$$\| \quad \mathcal{Q}_{S_{RT}}, \mathcal{CS}, lastRTsent, newRTGap, rtAtmpt := \mathcal{Q}_{S_{RT}} \bullet m, m, clock, rtGap, 0$$

$$\text{if } (host_{hid} \Gamma' host_{dstid})$$

$$\| \quad \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet r \quad \text{if } \neg(host_{hid} \Gamma' host_{dstid})$$

$$\rangle \quad \text{if } (\texttt{isMH}(dstid) \wedge (dstid \in assoc) \wedge (\mathcal{CS} = \perp))$$

$$\| \quad \langle \quad apid := \langle\ \exists\ e : (e \in \mathcal{H}) \wedge (e \uparrow 1 = dstid) :: e \uparrow 2\ \rangle$$

$$\| \quad \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, apid, RT, r, \mathrm{W}) \quad \text{if } (host_{hid} \Gamma' host_{apid})$$

$$\| \quad \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet r \quad \text{if } \neg(host_{hid} \Gamma' host_{apid})$$

$$\rangle \quad \text{if } (\texttt{isMH}(dstid) \wedge \neg(dstid \in assoc) \wedge \texttt{isPresent}_{\mathcal{H}}(dstid))$$

$$\| \quad \langle \ \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, B_{iBSS_{\mathrm{W}}}, Locate, dstid, \mathrm{W}) \ \| \ \mathcal{L} := \mathcal{L} \cup \{(dstid, r, clock)\}$$

$$\rangle \quad \text{if } (\texttt{isMH}(dstid) \wedge \neg(dstid \in assoc) \wedge \neg\texttt{isPresent}_{\mathcal{H}}(dstid))$$

$$\rangle$$

$$\texttt{commSend}_{\mathrm{MH}_{IBSS}}(\mathcal{Q}_{RT_S}) \triangleq$$

$$\langle \quad r, \mathcal{Q}_{RT_S} := \texttt{head}(\mathcal{Q}_{RT_S}), \texttt{tail}(\mathcal{Q}_{RT_S}) \quad \| \quad dstid := r \uparrow \mathrm{dstHost} \quad \| \quad m := \texttt{newMsg}(hid, dstid, RT, r, \mathrm{WL})$$

$$\| \quad \mathcal{Q}_{S_{RT}}, \mathcal{CS}, lastRTsent, newRTGap, rtAtmpt := \mathcal{Q}_{S_{RT}} \bullet m, m, clock, rtGap, 0$$

$$\text{if } (\texttt{isMH}(dstid) \wedge (host_{hid} \Gamma' host_{dstid}))$$

$$\| \quad \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet r \quad \text{if } (\neg\texttt{isMH}(dstid) \vee \neg(host_{hid} \Gamma' host_{dstid}))$$

$$\rangle \quad \text{if } ((status = connected) \wedge (\mathcal{CS} = \perp))$$

$$\texttt{commRcv}_{\mathrm{SH}_{iBSS}}(\mathcal{Q}_{R_{RT}}) \triangleq$$

$$\langle \quad m, \mathcal{Q}_{R_{RT}} := \texttt{head}(\mathcal{Q}_{R_{RT}}), \texttt{tail}(\mathcal{Q}_{R_{RT}})$$

$$\| \quad \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet m \cdot data \quad \text{if } (\texttt{isMsgRT}(m) \wedge (\texttt{isSH}(m \cdot src) \vee \texttt{isAP}(m \cdot src)) \wedge (m \cdot dest = hid))$$

$$\| \quad \langle \quad \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, m \cdot src, RT, \langle \exists e : (e \in \mathcal{L}) \wedge (e \uparrow 1 = m \cdot data) :: e \uparrow 2 \rangle, \mathrm{W})$$

$$\| \quad \mathcal{H} := \mathcal{H} \cup \{(m \cdot data, m \cdot src, clock)\}$$

$$\rangle \quad \text{if } (\texttt{isMsgFound}(m) \wedge \texttt{isAP}(m \cdot src) \wedge \texttt{isPresent}_{\mathcal{L}}(m \cdot data))$$

$$\rangle$$

$$\texttt{commRcv}_{\text{MH}_{iBSS}}(\mathcal{Q}_{R_{RT}}) \triangleq$$

$$\langle \quad m, \mathcal{Q}_{R_{RT}} := \texttt{head}(\mathcal{Q}_{R_{RT}}), \texttt{tail}(\mathcal{Q}_{R_{RT}})$$

$$\| \ \langle \ \langle \ \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet m.data \ \| \ \langle \exists e : (e \in \mathcal{L}RT) \wedge (e \uparrow 1 = m.src) :: e \uparrow 2 := m.mid \rangle$$

$$\rangle \quad \text{if } (\texttt{isPresent}_{\mathcal{L}RT}(m.src) \wedge \neg\texttt{isRepeat}_{\mathcal{L}RT}(m.src, m.mid))$$

$$\| \ \langle \ \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet m.data \ \| \ \mathcal{L}RT := \mathcal{L}RT \cup \{(m.src, m.mid)\} \ \rangle \quad \text{if } \neg\texttt{isPresent}_{\mathcal{L}RT}(m.src)$$

$$\| \ \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, m.src, ACK, m.mid, \text{WL})$$

$$\rangle \quad \text{if } (\texttt{isMsgRT}(m) \wedge \texttt{isAP}(m.src) \wedge (m.src = assoc[0]) \wedge (m.dest = hid))$$

$$\| \ \mathcal{CS} := \perp \quad \text{if } (\texttt{isMsgACK}(m) \wedge \texttt{isAP}(m.src) \wedge (m.src = assoc[0]) \wedge (m.dest = hid) \wedge (rtAtmpt < 3)$$

$$\wedge ((clock - lastRTsent) < newRTGap) \wedge \neg(\mathcal{CS} = \perp) \wedge (m.mid = \mathcal{CS}.mid))$$

$$\rangle$$

$$\texttt{commRcv}_{\text{AP}_{iBSS}}(\mathcal{Q}_{R_{RT}}) \triangleq$$

$$\langle \quad m, \mathcal{Q}_{R_{RT}} := \texttt{head}(\mathcal{Q}_{R_{RT}}), \texttt{tail}(\mathcal{Q}_{R_{RT}})$$

$$\| \ \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet m.data \quad \text{if } (\texttt{isMsgRT}(m) \wedge (\texttt{isSH}(m.src) \vee \texttt{isAP}(m.src)) \wedge (m.dest = hid))$$

$$\| \ \langle \ \langle \ \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet m.data \ \| \ \langle \exists e : (e \in \mathcal{L}RT) \wedge (e \uparrow 1 = m.src) :: e \uparrow 2 := m.mid \rangle$$

$$\rangle \quad \text{if } (\texttt{isPresent}_{\mathcal{L}RT}(m.src) \wedge \neg\texttt{isRepeat}_{\mathcal{L}RT}(m.src, m.data))$$

$$\| \ \langle \ \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet m.data \ \| \ \mathcal{L}RT := \mathcal{L}RT \cup \{(m.src, m.mid)\} \ \rangle \quad \text{if } \neg\texttt{isPresent}_{\mathcal{L}RT}(m.src)$$

$$\| \ \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, m.src, ACK, m.mid, \text{WL})$$

$$\rangle \quad \text{if } (\texttt{isMsgRT}(m) \wedge \texttt{isMH}(m.src) \wedge (m.src \in assoc) \wedge (m.dest = hid))$$

$$\| \ \mathcal{CS} := \perp \quad \text{if } (\texttt{isMsgACK}(m) \wedge \texttt{isMH}(m.src) \wedge (m.src \in assoc) \wedge (m.dest = hid) \wedge (rtAtmpt < 3)$$

$$\wedge ((clock - lastRTsent) < newRTGap) \wedge \neg(\mathcal{CS} = \perp) \wedge (m.mid = \mathcal{CS}.mid))$$

$$\| \ \langle \ \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, m.src, RT, \langle \exists e : (e \in \mathcal{L}) \wedge (e \uparrow 1 = m.data) :: e \uparrow 2 \rangle, \text{W})$$

$$\| \ \mathcal{H} := \mathcal{H} \cup \{(m.data, m.src, clock)\}$$

$$\rangle \quad \text{if } (\texttt{isMsgFound}(m) \wedge \texttt{isAP}(m.src) \wedge \texttt{isPresent}_{\mathcal{L}}(m.data))$$

$$\| \ \langle \ \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, m.src, Found, m.data, \text{W}) \quad \text{if } (m.data \in assoc)$$

$$\rangle \quad \text{if } (\texttt{isMsgLocate}(m) \wedge (\texttt{isSH}(m.src) \vee \texttt{isAP}(m.src)))$$

$$\rangle$$

$$M := \texttt{commReSend}_{\text{RT}_{iBSS}}() \triangleq$$

$$\langle \ M, lastRTsent, newRTGap, rtAtmpt := \mathcal{CS}, clock, (2 * newRTGap), (rtAtmpt + 1)$$

$$\text{if } (\neg(\mathcal{CS} = \perp) \wedge (rtAtmpt < 3))$$

$$\rangle$$

$$\texttt{commValid}_{\mathcal{H}\mathcal{L}_{iBSS}}() \triangleq$$

$$\langle \quad \langle \| \ e : e \in \mathcal{H} :: \mathcal{H} := \mathcal{H} \setminus \{e\} \quad \text{if } \neg\texttt{isValid}_{\mathcal{H}}(e, clock) \ \rangle$$

$$\| \ \langle \| \ e : e \in \mathcal{L} :: (\mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet e \uparrow 2) \ \| \ \mathcal{L} := \mathcal{L} \setminus \{e\} \quad \text{if } \neg\texttt{isValid}_{\mathcal{L}}(e, clock) \ \rangle$$

$$\rangle$$

$$M := \texttt{commReSend}_{\text{RT}_{IBSS}}() \triangleq$$

$$\langle \ M, lastRTsent, newRTGap, rtAtmpt := \mathcal{CS}, clock, (2 * newRTGap), (rtAtmpt + 1)$$

$$\text{if } (\neg(\mathcal{CS} = \perp) \wedge (rtAtmpt < 3)) \ \rangle$$

$$\texttt{commRcv}_{\text{MH}_{IBSS}}(\mathcal{Q}_{R_{RT}}) \triangleq$$

$$\langle \quad m, \mathcal{Q}_{R_{RT}} := \texttt{head}(\mathcal{Q}_{R_{RT}}), \texttt{tail}(\mathcal{Q}_{R_{RT}})$$

$$\| \ \langle \ \langle \ \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet m.data \ \| \ \langle \exists e : (e \in \mathcal{L}RT) \wedge (e \uparrow 1 = m.src) :: e \uparrow 2 := m.mid \rangle$$

$$\rangle \quad \text{if } (\texttt{isPresent}_{\mathcal{L}RT}(m.src) \wedge \neg\texttt{isRepeat}_{\mathcal{L}RT}(m.src, m.mid))$$

$$\| \ \langle \ \mathcal{Q}_{RT_R} := \mathcal{Q}_{RT_R} \bullet m.data \ \| \ \mathcal{L}RT := \mathcal{L}RT \cup \{(m.src, m.mid)\} \ \rangle \quad \text{if } \neg\texttt{isPresent}_{\mathcal{L}RT}(m.src)$$

$$\| \ \mathcal{Q}_{S_{RT}} := \mathcal{Q}_{S_{RT}} \bullet \texttt{newMsg}(hid, m.src, ACK, m.mid, \text{WL})$$

$$\rangle \quad \text{if } (\texttt{isMsgRT}(m) \wedge \texttt{isMH}(m.src) \wedge (m.dest = hid))$$

$$\| \ \mathcal{CS} := \perp \quad \text{if } (\texttt{isMsgACK}(m) \wedge \texttt{isMH}(m.src) \wedge (m.dest = hid) \wedge (rtAtmpt < 3)$$

$$\wedge ((clock - lastRTsent) < newRTGap) \wedge \neg(\mathcal{CS} = \perp) \wedge (m.mid = \mathcal{CS}.mid))$$

$$\rangle$$