Mobile Information Systems 4 (2008) 333–349
IOS Press

Novel directory service and message delivery mechanism enabling scalable mobile agent communication

Jinho Ahn*
Department of Computer Science, Kyonggi University, Suwon, Korea

Abstract. Mobile agent technology has emerged as a promising programming paradigm for developing highly dynamic and large-scale service-oriented computing middlewares due to its desirable features. For this purpose, first of all, scalable location-transparent agent communication issue should be addressed in mobile agent systems despite agent mobility. Although there were proposed several directory service and message delivery mechanisms, their disadvantages force them not to be appropriate to both low-overhead location management and fast delivery of messages to agents migrating frequently. To mitigate their limitations, this paper presents a scalable distributed directory service and message delivery mechanism. The proposed mechanism enables each mobile agent to autonomously leave tails of forwarding pointers on some few of its visiting nodes depending on its preferences. This feature results in low message forwarding overhead and low storage and maintenance cost of increasing chains of pointers per host. Also, keeping mobile agent location information in the effective binding cache of each sending agent, the sending agent can communicate with mobile agents much faster compared with the existing ones.

Keywords: Distributed system, mobile agent, directory service, message delivery, scalability

1. Introduction

Thanks to its desirable features such as asynchronous and dynamic execution and autonomy [12,17, 22,25], mobile agent technology has emerged as a promising programming paradigm for developing highly dynamic and large-scale service-oriented computing middlewares in various fields like ubiquitous computing [1,3,5,9,10,26], web service [28], grid computing [13,15] and adhoc network [19]. However, as mobile agent systems become built on large-scale networks and the number of mobile agents is rapidly increasing, several research issues related to mobile agents should be reconsidered to satisfy this demand. Among them, it is most important to enhance the performance of the agent communication in large-scale infrastructures. For this purpose, some effective and efficient inter-agent communication mechanism is required in distributed agent-based systems. Agent mobility may lead to the loss of messages being destined to an agent on its migration. Thus, it causes reliable inter-agent communications to be not easy to achieve in the distributed agent based systems. Especially, guaranteeing the delivery of messages to highly mobile agents, which move frequently among service nodes, is a more challenging problem. In this paper, we attempt to address these issues effectively like in Fig. 1. Although there were proposed

*Address for correspondence: Department of C.S., Kyonggi University, San 94-6 Yiuidong, Yeongtonggu, Suwonsi Gyeonggido 443-760, Korea. Tel.: +82 31 249-9674; Fax: +82 31 249-9673; E-mail: jhahn@kyonggi.ac.kr.

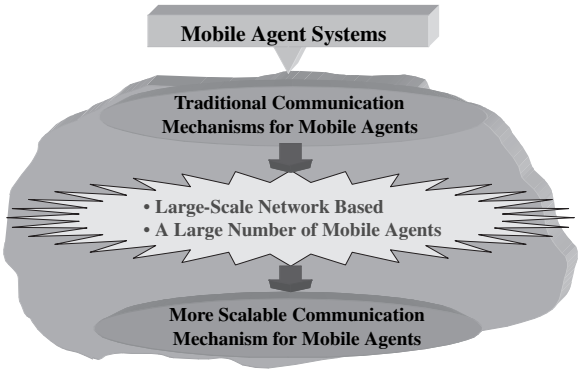

Fig. 1. The motivation of our work.

several directory service and message delivery mechanisms to address this issue as mentioned below, their disadvantages force them not to be appropriate to both low-overhead location management and fast delivery of messages to agents migrating frequently.

Firstly, the broadcast-based mechanism [18] guarantees transparent and reliable inter-agent communication and can also provide multicast communication to a set of agents. But, to locate the message destination, the mechanism has to contact every visiting node in the network. Thus, its large traffic overhead makes broadcasts impractical in large-scale distributed agent systems.

Secondly, in the home-based mechanism [14,21], every mobile agent has a home node and should register its current location with the home node whenever it moves. Thus, when some messages are sent to a mobile agent currently located at a foreign node, the messages are first directed to its home node, which forwards them to the agent. This mechanism is simple to implement and results in little mobile agent locating overhead. However, it is unsuitable for highly mobile agents in distributed agent based systems because every agent location updating and message delivery are all performed around the home node, which introduces centralization. Additionally, in the distributed agent-based systems, the home node may be disconnected from the network.

Lastly, the forwarding pointer-based mechanism [11,16] forces each node on a mobile agent's movement path to keep a forwarding pointer to the next node on the path. Thus, if a message is delivered to an agent not being at the home node, the message must traverse a list of forwarding nodes. Thus, this mechanism can avoid performance bottlenecks of the global infrastructure, and therefore improve its scalability, particularly in large-scale distributed agent-based systems, compared with the home based one. Additionally, even if a home node is disconnected from the rest of the network, the forwarding pointer based mechanism allows agents registering with the node to communicate with other agents. However, as highly mobile agents leads to the length of their chains of pointers being rapidly increasing, its message forwarding overhead may be significantly larger. Furthermore, the number of forwarding pointers each service node needs to keep on its storage may exponentially increase if a large number of mobile agents are running in the systems. In a previous work [16], a type of update message called $inform$ message was introduced to include an agent's current location for shortening the length of trails of forwarding pointers. In this case, a node that receives the message is allowed to update its table if the received information is more recent than the one it had. However, it introduces no concrete and efficient solutions for this purpose, for example, when update messages should be sent, and which node they should be sent to. In order to mitigate their limitations, this paper presents a scalable distributed directory service and message delivery mechanism. The proposed mechanism enables each mobile agent

to autonomously leave tails of forwarding pointers on some few of its visiting nodes depending on its preferences. This feature results in low message forwarding overhead and low storage and maintenance cost of increasing chains of pointers per host. Also, keeping mobile agent location information in the effective binding cache of each sending agent, the sending agent can communicate with mobile agents much faster compared with the existing ones.

The rest of the paper is organized as follows. Section 2 describes our mobile agent system model assumed in this paper. In Section 3, we present our directory service and message delivery mechanism and prove its correctness. Section 4 makes several experiments for performance evaluation and Section 5 reviews related work. Finally, Section 6 concludes this paper.

2. Mobile agent system model

In this paper, we consider an asynchronous distributed agent based system where there is no global memory, no global clock and no bound on message delay. The system consists of a set of agent service nodes. Each service node supports an environment in which agents can operate safely and securely, and provides a uniform set of services that visiting agents can access its local resources in a limited way regardless of their locations. An agent is initially created on a service node, called $home\ node$ of the agent, and is given a unique identifier within the node. So, each agent can be identified as a globally unique object in the system by using the combination of its local identifier and the identifier of its home node. When an agent migrates in the system, its code and state information are captured and then transferred to the next node. After arriving at the node, the mobile agent resumes and performs its task, if needed, by interacting with other agents. In order to perform an assigned task on behalf of a user, a mobile agent ma executes on a sequence of $l(l > 1)$ service nodes $I_{ma} = [N_{home}, N_1, \cdots, N_{(l-1)}]$ according to its itinerary, which may be statically determined before the mobile agent is launched at its home node or dynamically while progressing its execution. It is assumed that communication channels support standard asynchronous message passing and are immune to partitioning, and reliable and FIFO. Mobile agents can migrate and messages be passed along these channels. Finally, we assume that each service node is failure-free because this makes it more easy to design and understand our agent tracking and message delivery mechanism explained hereafter.

3. Scalable directory service and message delivery mechanism

In this section, we first explain basic concept of our scalable mobile agent communication mechanism, describe its two components, directory service and message delivery algorithms in detail respectively, and then prove the correctness of the mechanism.

3.1. Basic idea

We attempt to develop our directory service and message delivery mechanism to have all three desirable features compared with the others as follows.

+ Alleviate the centralized dependency on the home node and its disconnection problem.
+ Require a small size of storage of each service node for agent location management.
+ Incur low message forwarding overhead.

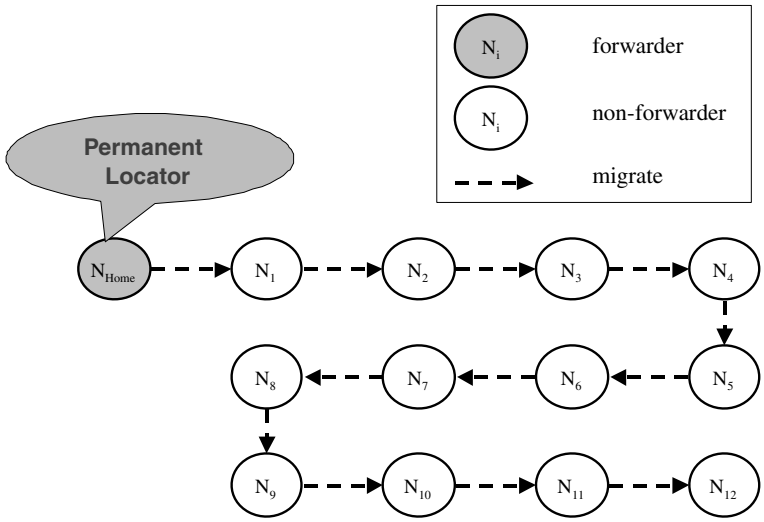

Fig. 2. Home-based mechanism.

Before discussing, the definition of two special directory service nodes should be explained. First, forwarder of an agent means a service node keeping a forwarding pointer of the agent on its storage. Thus, depending on the behavior of agent communication mechanisms, there may exist various number of forwarders of each agent in the system. Second, locator of an agent is the forwarder managing the identifier of the node where the agent is currently located. In this paper, it is assumed that there is only one locator in the system. We will explain how the proposed mechanism satisfies the three requirements using Figs 2, 3 and 4. These figures show all the same example that a mobile agent ma performs its task while migrating from its home node to nodes N_1 through N_{12} in node number order. In home-based mechanism, N_{home} among all visiting nodes $\in I_{ma}$ is the only forwarder of agent ma i.e., its permanent locator like in Fig. 2. Thus, this mechanism forces every agent created on a node to update its current location information only at the home node on every migration. Also, all messages destined to the agent must be always delivered to the agent only via its home node. Therefore, the mechanism doesn't address the first requirement.

In forwarding pointer-based mechanism, every visiting node $\in I_{ma}$ is a forwarder of agent ma like in Fig. 3. Thus, when each agent moves to another node, the agent has only to register its location only with the right before visiting node. Additionally, each sent message can be transmitted to its target agent via the home node as well as other forwarders of the agent. Therefore, the mechanism can satisfy the first requirement. However, in this mechanism, whenever each agent, especially highly mobile agent, migrates to a new node, its previous node has to keep a forwarding pointer of the agent. This feature may lead to significantly increasing the size of storage on which each service node maintains location information of agents. Moreover, as the forwarding path for delivering each message to its target agent highly increases due to the feature like in Fig. 3, its delivery time also increases at the same rate. Therefore, neither the second requirement nor the third one is addressed in this mechanism.

To satisfy all the three requirements, our proposed mechanism enables each mobile agent to autonomously and systematically leave trails of forwarding pointers only on some few of its visiting nodes depending on its preferences such as location updating and message delivery costs, network topology, security policies, communication patterns and so on, where the methods for selecting suitable forwarders of each mobile agent based on its administrator's policy may be embedded in the software logic of the

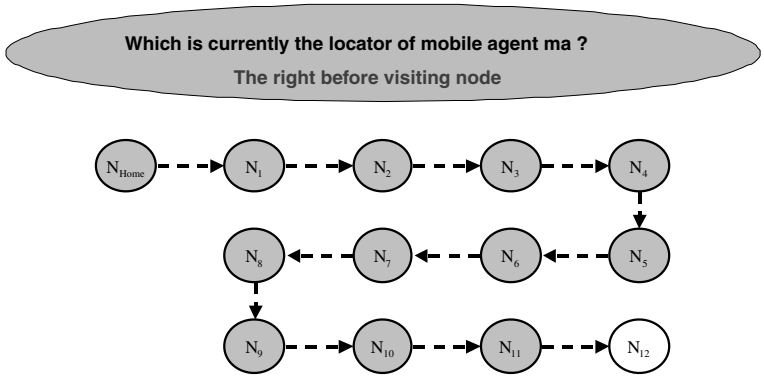

Fig. 3. Forwarding pointer-based mechanism.

agent. This behavior can considerably reduce both the amount of agent location information each node needs to maintain and the delivery time of each message because the length of its forwarding path may be much more shortened. Also, since there exist multiple forwarders, not only one, in this mechanism, the home node centralization can be avoided. But, as a part of the visiting nodes are forwarders in this mechanism, a new method is required to consistently manage location of each agent and enable each sent message to be delivered to the agent despite its migrations unlike in the previous forwarding pointer-based one. To satisfy the goal, the method is performed as follows. In Fig. 4, when agent ma moves from N_{home} to N_1, it forces N_1 to be its forwarder. In this case, N_1 becomes ma's locator. Afterwards in this method, ma should register its location with N_1 on every migration until it moves to the next forwarder N_5. When the agent arrives at N_5, N_5 informs N_1 that N_5 plays a role of ma's locator from now on. Hereafter, ma's current location information is managed by N_5 until the agent arrives at its next forwarder N_{10}. This method may result in slightly higher location update cost on every agent migration than that of the previous forwarding pointer-based mechanism. However, if each agent chooses some among its visiting nodes as forwarders by properly considering several important performance factors, the gap between the two costs may be almost negligible.

3.2. Data structures and algorithms

3.2.1. Directory service

Every node N_i should keep the following data structures in the directory service algorithm.

• $MAExecuted_i$: A table for saving location information of every agent currently running on N_i. Its element is a tuple (AID, LID, ts, $mFlag$, $msgQ$). LID is the identifier of agent AID's locator. ts is the timestamp associated with agent AID when the agent is located at N_i. Its value is incremented by one every time the corresponding agent migrates. Thus, when agent AID migrates to N_i, N_i should inform LID of both its identifier and ts so that LID can locate the agent. $mFlag$ is a bit flag designating if the agent is currently migrating to another node($= 1$) or not($= 0$). $msgQ$ is a message queue for buffering all the messages destined to agent AID on its migration. Its element consists of two fields, a message m and the source node ID of the message, SID.

• $FPointers_i$: A table for saving location information of every mobile agent which is not currently running on N_i, but of which N_i is a forwarder. Its element is a tuple (AID, NID, ts, $lFlag$, $mFlag$,

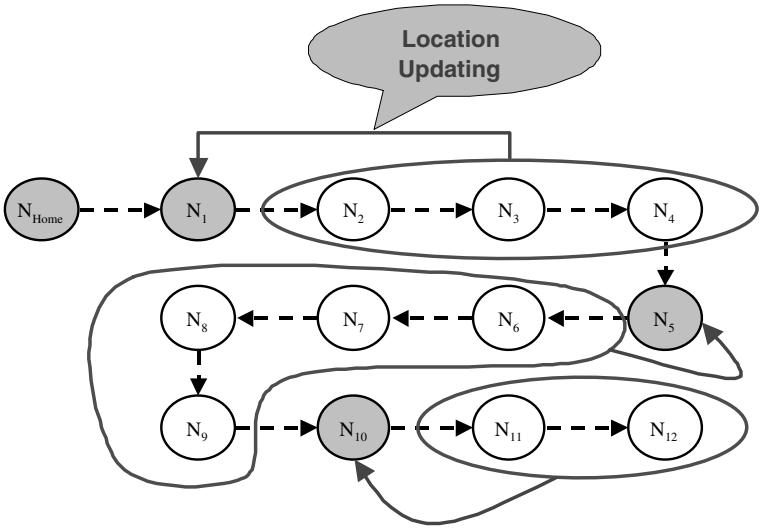

Fig. 4. Our scalable mechanism.

$msgQ$). NID is the identifier of the node where N_i knows agent AID is currently located and running. ts is the timestamp associated with the agent when the agent is located at node N_{NID}. It is used for avoiding updating recent location information by older information [16]. $lFlag$ is a bit flag indicating whether N_i is agent AID's locator or not. In the first case, its value is 1 and otherwise, 0. $mFlag$ is a bit flag designating if the agent is currently migrating to another node(=1) or not(=0). Like in $MAExecuted_i$, $msgQ$ is a message queue for buffering all the messages destined to agent AID on its migration. Its element consists of two fields, a message m and the source node ID of the message SID.

The algorithm for managing each agent's location on its migration is informally described using Fig. 5. This figure shows message interactions between nodes occurring in ma's location updating while migrating from its home node to N_1 through N_5. In Fig. 5, ma is created on N_{home} and then an element for ma (ID_{ma}, $home$, 0, 0, $\emptyset$) is saved into $MAExecuted_{home}$. If ma attempts to move to N_1 after having performed its partial task in the first step, it inserts into $FPointers_{home}$ ma's element (ID_{ma}, 1, 1, 1, 1, $\emptyset$) indicating N_{home} is ma's locator and ma is currently moving to N_1. Then, N_{home} sends a message $Migrated(ma)$ to N_1 for dispatching the agent with the identifier of the first and ma's timestamp into the latter. When receiving these, N_1 increases the timestamp by one. In this case, as ma wants N_1 to be its locator, it inserts ma's location information (ID_{ma}, 1, 1, 1, $\emptyset$) into $MAExecuted_1$. In the second step, N_1 sends N_{home} a message $Replaced(ID_{ma}, 1)$ including ma's timestamp in order to inform N_{home} that N_1 is ma's locator from now. On receiving the message, N_{home} updates ma's location information in $FPointers_{home}$ using the message and sets two fields of ma's element, $lFlag$ and $mFlag$, both to 0. If the messages destined to ma have been buffered in N_{home}'s message queue due to the migration, they are transmitted to N_1. When ma attempts to migrate to N_2 after ma has performed a part of its task in the third step, N_1 puts ma's element (ID_{ma}, 2, 2, 1, 1, $\emptyset$) into $FPointers_1$ and then sends a message $Migrated(ma)$ for moving agent ma to N_2. In this case, N_2 increments ma's timestamp by one and then inserts ma's element (ID_{ma}, 1, 2, 1, $\emptyset$) into $MAExecuted_2$ because ma wants N_2 to be just a visiting node. In the fourth step, the node sends a message $Register(ID_{ma}, 2)$ including the timestamp to its locator N_1 for registering ma's current location with the locator. If there are any messages sent to the agent in the queue of N_1, they are forwarded to N_2.

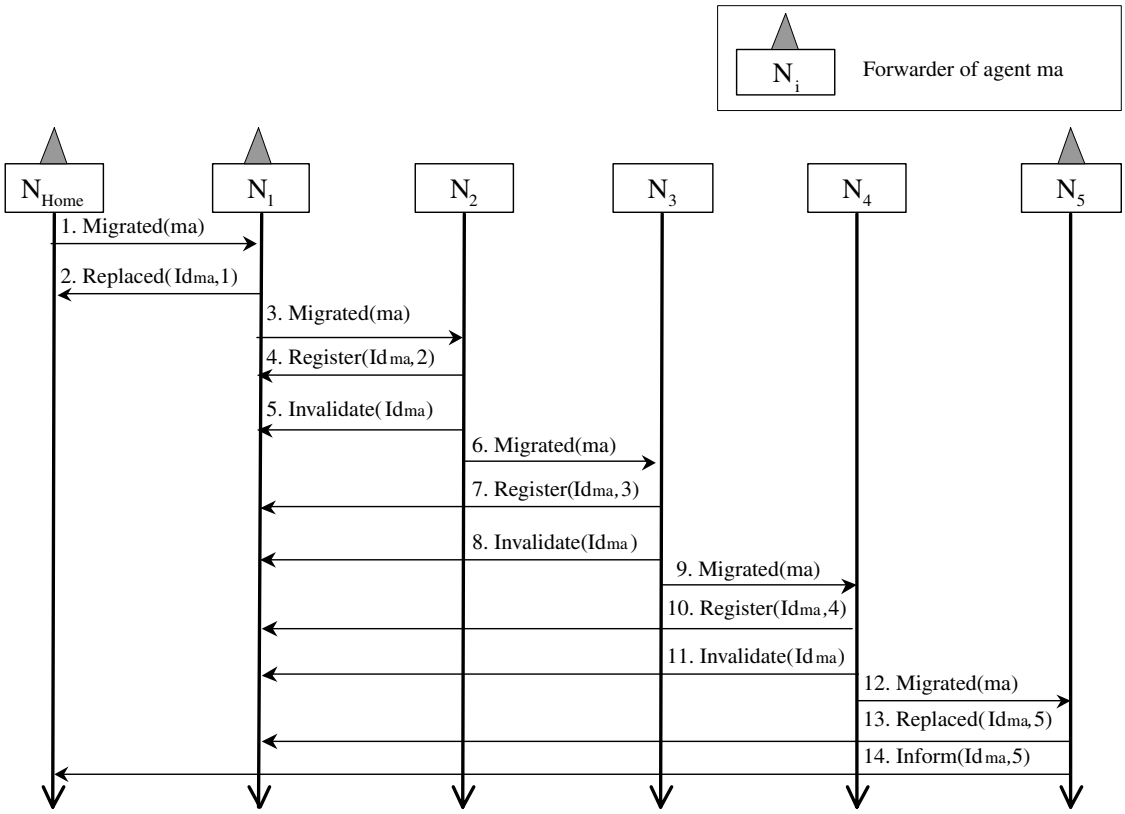

Fig. 5. An execution of our proposed directory service algorithm for an agent ma during its multi-node movement.

In the fifth step, N_2 first sends N_1 a message $Invalidate(ID_{ma})$ indicating that ma's migration process begins from now. When receiving the message, N_1 sets one field of ma's element, $mFlag$, to 1. Suppose the migration is started without the execution of this invalidation procedure. If N_1 receives any message destined to ma in this case, it forwards the message to N_2 because it doesn't know whether the migration continues to be executed. But, neither ma may be currently running on N_2 nor N_2 keep ma's location information on $FPointers_2$ because N_2 isn't ma's forwarder. In this case, the message cannot be delivered to ma. Therefore, N_2 should transmit a message $Migrated(ma)$ to N_3 after having performed the invalidation procedure on ma's locator, and then remove ma's element from $MAExecuted_2$. Afterwards, ma's visiting node N_3 increments ma's timestamp and saves ma's element (ID_{ma}, 1, 3, 1, $\emptyset$) into $MAExecuted_3$ in the sixth step, and then sends a message $Register(ID_{ma}, 3)$ to N_1 in the seventh step. On the receipt of the message, N_1 updates ma's element in $FPointers_1$ to (ID_{ma}, 3, 3, 1, 0, $\emptyset$) using the message. When ma migrates to N_4, the same procedure is performed like in this figure. When agent ma moves from N_4 to N_5 and decides that N_5 is the proper node as ma's locator, N_5 creates ma's location information (ID_{ma}, 5, 5, 1, $\emptyset$) and inserts it into $MAExecuted_5$ in the twelfth step. Afterwards, in the thirteenth step, the node sends N_1 a message $Replaced(ID_{ma}, 5)$ for notifying the previous locator N_1 that N_5 is ma's locator from now. Also, ma may send a message $Inform(ID_{ma}, 5)$ to N_{home} for updating its current location at N_{home} in the fourteenth step to reduce the message delivery time incurred when another agent initially sends a message to ma via N_{home}. If ma recognizes this consideration helps no performance improvement, it doesn't perform the update procedure on the home node.

Module MIGRATE-TO($ma, targetNodeID$)
  find the element $comp$ of agent ma from $MAExecuted_i$;
  $comp.mFlag ::= 1$
  if ($comp.LID$ is equal to i) **then**
    $FPointers_i ::= FPointers_i \bigcup$
      $\{(comp.AID, targetNodeID, comp.ts{+}1, 1, 1)\}$;
  else
    remote call Module INVALIDATE($comp.AID$) **at** node $comp.LID$;
  remote call Module MIGRATED(ma, $comp$) **at** node $targetNodeID$;
  $MAExecuted_i ::= MAExecuted_i - \{comp\}$;

Module MIGRATED($ma, comp$)
  $comp.ts{+}{+}$;
  if ($comp.LID$ is not equal to i) **then**
    if (node i is a forwarder of the agent ma or should become its locator) **then** {
      $FID ::= comp.LID$; $comp.LID ::= i$;
      $msgQ$ ::= **remote call Module** REPLACED($comp.AID$, i, $comp.ts$) **at** node FID ;
      $comp.msgQ ::= comp.msgQ \bigcup msgQ$;
      if (FID is not equal to ID of ma's home node and
          this locator replacement should be informed the home node) **then**
        remote call Module INFORM($comp.AID$, i, $comp.ts$) **at** ma's home node ;
    } **else** {
      $msgQ$ ::= **remote call Module** REGISTER($comp.AID$, i, $comp.ts$) **at** node $comp.LID$;
      $comp.msgQ ::= comp.msgQ \bigcup msgQ$;
    }
  if ($\exists$*loc-info* $\in FPointers_i$ **st** *loc-info*$.AID == comp.AID$) **then**
    $FPointers_i ::= FPointers_i - \{$*loc-info*$\}$;
  $MAExecuted_i ::= MAExecuted_i \bigcup \{comp\}$;
  resume ma **with** $comp.msgQ$;
  $comp.mFlag ::= 0$

Module INVALIDATE($agentID$)
  find the element $comp$ of agent $agentID$ from $FPointers_i$;
  $comp.mFlag ::= 1$;

Fig. 6. Modules for a service node i on agent migration.

The informally stated modules for every service node in the directory service algorithm are formally given in Figs 6 and 7.

3.2.2. Message delivery

Every node N_i should have an agent location cache $MALinks_i$ in the message delivery algorithm. It is a cache for temporarily storing location information of each mobile agent which agents running on N_i communicate with. Its element is a tuple (AID, NID, ts). NID is the identifier of the node where N_i knows agent AID is currently located and running. Thus, when attempting to deliver messages to agent AID, each agent on N_i forwards them to NID regardless of whether this address is outdated. ts is the timestamp assigned to agent AID when the agent was located at node N_{NID}.

We intend to use an example in Fig. 8 to clarify the algorithm to enable every sent message to be reliably delivered to its target agent despite agent migrations. This example illustrates agent γ sends three messages, m_1, m_2 and m_3 to agent ma in this order while ma is migrating from its home node to N_1 through N_5 according to its itinerary. In this figure, after ma has moved from N_{home} to N_3, γ at N_w attempts to deliver the first message m_1 to ma. In this case, N_w has no location information for ma in its location cache $MALinks_w$. Thus, N_w creates and saves ma's element (ID_{ma}, $home$, 0) into

Module REGISTER($agentID$, $currentNodeID$, $timestamp$)
find the element $comp$ of agent $agentID$ from $FPointers_i$;
$comp.NID ::= currentNodeID$; $comp.ts ::= timestamp$;
$comp.mFlag ::= 0$; **return** $comp.msgQ$;

Module INFORM($agentID$, $locatorID$, $timestamp$)
find the element $comp$ of agent $agentID$ from $FPointers_i$;
$comp.NID ::= locatorID$; $comp.ts ::= timestamp$;

Module REPLACED($agentID$, $locatorID$, $timestamp$)
find the element $comp$ of agent $agentID$ from $FPointers_i$;
$comp.NID ::= locatorID$; $comp.ts ::= timestamp$;
$comp.lFlag ::= 0$; $comp.mFlag ::= 0$;
return $comp.msgQ$;

Fig. 7. Modules for a service node i on agent migration (continued).

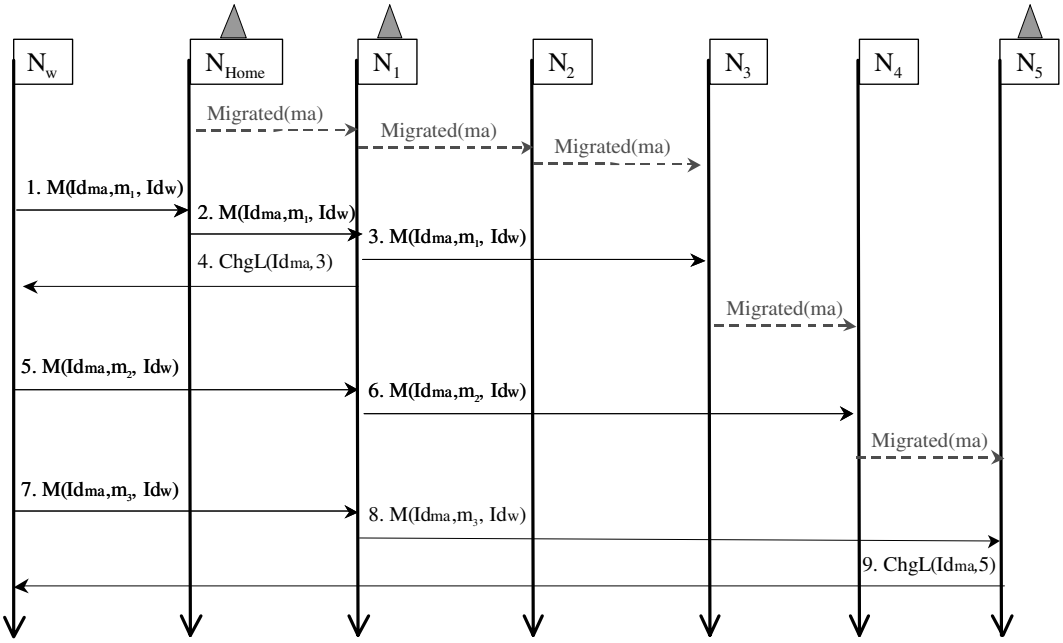

Fig. 8. An execution of reliably forwarding messages from their source agent γ to their target agent ma during its multi-node movement.

$MALinks_w$. Then, it sends the message m_1 to N_{home}. Receiving the message, N_{home} retrieves ma's element from $FPointers_{home}$. In this case, as the value of the bit flag $lFlag$ in the element is 0, N_{home} isn't ma's locator. Thus, it consults the element and forwards the message m_1 to the next forwarder N_1. On the receipt of the message, N_1 obtains ma's element from $FPointers_1$ and then checks the flag $lFlag$ in the element. In this case, N_1 is ma's locator because the value of the flag is 1. Also, as the value of the second flag $mFlag$ is 0, it forwards the message to ma's currently running node N_3 by consulting the element. At the same time, as N_w has the outdated identifier of ma's locator, N_1 sends N_w a message $ChgL(ID_{ma}, 3)$ containing the identifier of ma's current locator(=N_1) and timestamp(=3) in the figure. When receiving the message, N_w updates ma's element in $MALinks_w$ using the message.

```
Module SENDM(AID, m, LID)
    if (∃loc-info ∈ MAExecuted_i st loc-info.AID == AID) then
      if (loc-info.mFlag == 0) then  deliver m to agent AID ;
      else  loc-info.msgQ ::= loc-info.msgQ ⋃ {(m, SID)} ;
    else if(∃comp ∈ MALinks_i st comp.AID == AID) then  FID ::= comp.NID ;
    else {
      FID ::= ID of agent AID's home node ;
      MALinks_i ::= MALinks_i ⋃ {(AID,FID, 0)} ;
    }
    transmit a message M(AID, m, i) to node FID ;

// On receive a message M()
Module RECEIVEM(AID, m, SID)
    if (∃loc-info ∈ MAExecuted_i st loc-info.AID == AID) then
      if (loc-info.mFlag == 0) then  deliver m to agent AID ;
      else  loc-info.msgQ ::= loc-info.msgQ ⋃ {(m, SID)} ;
    else if(∃comp ∈ FPointers_i st comp.AID == AID) then
      if (comp.lFlag == 1) then
        if (comp.mFlag == 1) then
          comp.msgQ ::= comp.msgQ ⋃ {(m, SID)} ;
        else
          transmit a message M(AID, m, SID) to node comp.NID ;
      else
        transmit a message MFromFwdr(AID, m, SID) to node comp.NID ;
```

Fig. 9. Modules for a service node i on message delivery.

When ma migrates from N_3 to N_4 and then γ at N_w sends the second message m_2 to ma, N_w finds ma's element from $MALinks_w$ and then forwards m_2 to N_1. On the receipt of m_2, N_1 can see that it is ma's current locator because the value of the bit flag $lFlag$ of a's element in $FPointers_1$ is 1. Also, as the value of the flag $mFlag$ is 0, N_1 can send m_2 to ma's currently running node($= N_4$). In this case, as γ knows the identifier of ma's current locator, N_1 doesn't send any message $ChgL$ to N_w.

After ma has migrated from N_4 to N_5, γ transmits the third message m_3 to ma. At this point, N_w sends m_3 to N_1 by consulting ma's element in $MALinks_w$. In this case, as the value of the flag $lFlag$ of ma's element in $FPointers_1$ is 0, N_1 forwards m_3 to the next forwarder N_5. When N_5 receives the message, it can deliver the message to N_5 because it recognizes that it is ma's locator and ma is currently running on it. Concurrently, N_5 informs N_w that ma's current locator is N_5 by sending a message $ChgL(ID_{ma}, 5)$ to N_w.

The modules for every service node in the message delivery algorithm are formally described in Figs 9 and 10.

3.3. Correctness

In this section, we prove the correctness of our proposed scalable communication mechanism.

Definition 1. $ForwardingPath(msg, ID_{ma}, N_i)$ is a sequence $< N_i, ..., N_n >$ of a mobile agent ma's forwarders which compose an acyclic path for delivering a message msg from N_i to N_n, the current locator of agent ma. For all N_j,N_k in the path, if $i \leqslant j < k \leqslant n$, agent ma visited N_j and then N_k.

Theorem 1. When any message is sent to an agent in the system, the proposed mechanism enables the message to correctly be delivered to the target agent despite its migrations.

```
// On receive a message MFromFwdr()
 Module RECEIVEMFROMFWRDR(AID, m, SID)
      if (∃loc-info ∈ MAExecuted_i st loc-info.AID == AID) then
        if (loc-info.mFlag == 0) then   deliver m to agent AID ;
        else   loc-info.msgQ ::= loc-info.msgQ ⋃ {(m, SID)} ;
        transmit a message chgL(AID, loc-info.ts, loc-info.LID) to node SID ;
      else if(∃comp ∈ FPointers_i st comp.AID == AID) then
        if (comp.lFlag == 1) then
          if (comp.mFlag == 1) then
            comp.msgQ ::= comp.msgQ ⋃ {(m, SID)} ;
          else
            transmit a message M(AID, m, SID) to node comp.NID ;
          transmit a message chgL(AID, comp.ts, i) to node SID ;
        } else
            transmit a message MFromFwdr(AID, m, SID) to node comp.NID ;

// On receive a message chgL()
 Module CHANGELINK(AID, timestamp, LID)
      find the element comp of agent agentID from MALinks_i ;
      if (comp.ts is less than timestamp) then {
        comp.NID ::= LID ;   comp.ts ::= timestamp ;
      }
```

Fig. 10. Modules for a service node i on message delivery (continued).

Proof. Assume that a message msg has to be delivered to its target agent ma. In this case, the sender agent of the message is γ at node N_i and agent ma is currently running on node N_j. When N_i attempts to send the message to agent ma, N_i first looks up the address for ma from its agent location cache $MALinks_i$. If there is the address in the cache, N_i sends the message to this address. Otherwise, it obtains the address of ma's home node from ma's identifier and then sends the message to the address. In both cases, suppose the forwarding address is denoted by N_l. The proof proceeds by induction on the number of all the forwarders $\in ForwardingPath(msg, ID_{ma}, N_l)$, denoted by $|ForwardingPath(msg, ID_{ma}, N_l)|$.

[Base case]

In this case, N_l is the locator of agent ma. Therefore, the following two cases should be considered.

Case 1: $l = j$.

In this case, N_j can find ma's location information e from $MAExecuted_j$. There are two subcases considered.

Case 1.1: $e.mFlag = 0$.

In this case, the message msg is trivially delivered to ma because ma is currently running on node N_j.

Case 1.2: $e.mFlag = 1$.

In this case, agent ma attempts to move to another node N_{j+1}. Thus, the message is saved on ma's message queue $e.msgQ$ in $MAExecuted_j$ until the movement process is completed. After agent ma with the element e including the message has been migrated to N_{j+1}, the message can be correctly delivered to ma when the agent resumes its execution on the node N_{j+1}.

Case 2: $l < j$.

In this case, N_l looks up ma's location information e from $FPointers_l$. Then, it checks the value of a flag variable $e.mFlag$. There are two subcases according to the value.

Case 2.1: $e.mFlag = 0$.

In this case, the message is sent directly to N_j. Then, the subsequent procedure is performed like in case 1. Therefore, ma can correctly receive the message.

Case 2.2: $e.mFlag = 1$.

In this case, agent ma is migrating to another node N_{j+1}. Thus, the message is buffered into ma's message queue $e.msgQ$ in $FPointers_l$ until the movement process is completed. After the migration, the value of a flag variable $e.mFlag$ changes to 0 and then the message recorded in the queue is forwarded to N_{j+1}. Therefore, the message can be correctly delivered to ma.

[Induction hypothesis]

We assume that the theorem is true for the message msg in case that $|ForwardingPath(msg, ID_{ma}, N_l)| = k$.

[Induction step]

After the message has been safely routed to the k-th forwarder N_α by induction hypothesis, N_α retrieves ma's location information e from $FPointers_\alpha$. Then, it forwards the message to the (k+1)-th forwarder N_β, which is the locator of agent ma. The following case is similar to the base case mentioned above. Therefore, the message can be correctly delivered to ma.

By induction, the mechanism can correctly deliver any sent message msg to its corresponding agent even if the agent migrates. □

4. Experimental evaluation

In this section, we compare performance of our proposed mechanism($Proposed$) with that of an existing forwarding pointer-based one($Existing$) [16] by performing an extensive simulation using a discrete-event simulation language [2]. There are two performance indices as follows. The first is Hop_{Cntr}, the average number of nodes which have forwarded a message destined to a mobile agent until the message is actually delivered to the agent. The second is $LInfo_{no}$, the average number of agent location entries that each service node keeps on its storage.

A simulated system is composed of 100 nodes each associated with a coordinate(x, y). They are all interconnected with each other. Any two adjacent nodes are connected with a LAN link having a bandwidth of 10Mbps and a propagation delay of 2ms. For simplicity of this simulation, we assume that both bandwidth and propagation delay between any pair of nodes are proportional to their distance. In our simulation environment, there are some important simulation parameters as follows. The first is Set_{snd}, the set of nodes being capable of sending messages to a mobile agent. Nodes in the set are uniformly distributed along the entire network. The second is $LChg_{thrd}$, the threshold which is required when a mobile agent decides whether the current locator of the agent should be changed. If a randomly generated probability variable δ for an agent ma is greater than its threshold $LChg_{thrd}$ when the agent attempts to migrate to another node, the next node becomes the locator of the agent. In this simulation, it is assumed that the threshold $LChg_{thrd}$ is determined when its agent is created and isn't changed during the entire life cycle of its agent any more. The third is $Stay_{time}$, the mean time elapsed when a mobile agent stays at a node for performing its task, following an exponential distribution. The last simulation parameter is $ASpawned_{no}$, the total number of agents created in the system. Also, the size of each application message transmitted between any two agents ranges from 512bytes to 1Kbytes and the size of each control message for the inter-agent communication is 128bytes. All experimental results shown in this simulation are all averages over a number of trials.

Figure 11 shows the average message forwarding overhead, Hop_{Cntr}, for the two directory service and message delivery mechanisms for the specified range of the $Stay_{time}$. In this figure, as the $Stay_{time}$ of

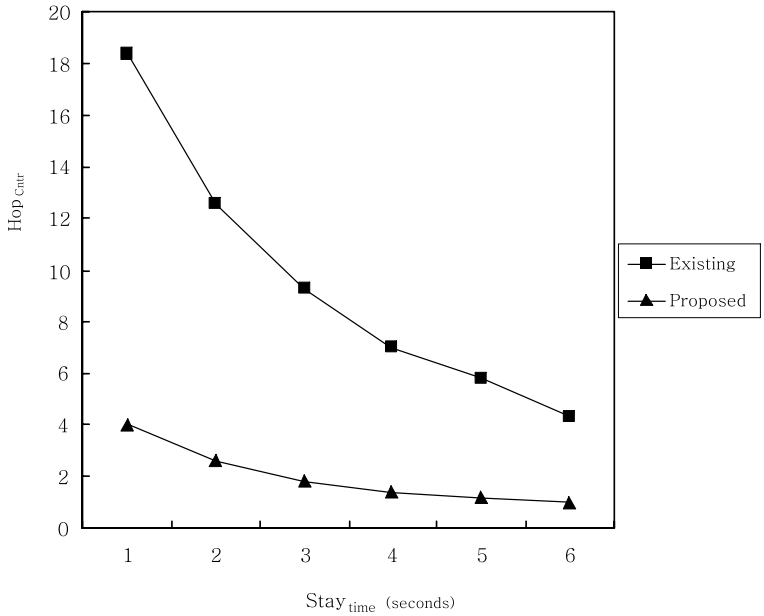

Fig. 11. Hop_{Cntr} vs. $Stay_{time}$.

the two mechanisms decreases, their Hop_{Cntr}s increase. The reason is that as the length of each agent's stay grows shorter, the path for forwarding messages to the agent is contrarily made longer. However, we can see that the Hop_{Cntr} of $Proposed$ is significantly lower than that of $Existing$ in this figure. In particular, as the $Stay_{time}$ becomes shorter, the gap between their message delivery costs considerably grows larger. In these results, the mechanism $Proposed$ reduces about 76–79% of Hop_{Cntr} compared with the mechanism $Existing$.

Figure 12 shows the amount of agent location information kept by each service node, $LInfo_{no}$, for the two mechanisms with varying the number of agents generated in the simulated system, $ASpawned_{no}$. As shown in this figure, increasing the $ASpawned_{no}$ of the two mechanisms causes their $LInfo_{no}$s to grow higher accordingly. This is because as the number of agents created in the system increases, the two mechanisms are stepping up the amount of agent location information maintained by each node. But, as expected, $Proposed$ provides very high performance gains in $LInfo_{no}$ over $Existing$. Especially, when $ASpawned_{no}$ is increasing, the increasing rate of $LInfo_{no}$ of $Proposed$ is rising much slower than that of $Existing$. In this simulation, the mechanism $Proposed$ reduces $LInfo_{no}$ by 79% compared with the mechanism $Existing$.

Therefore, these simulation results show that even if the scale of a mobile agent system grows considerably, our proposed mechanism $Proposed$ incurs low message forwarding overhead and reduces the delivery time of each message destined to a mobile agent and the amount of agent location information kept by each service node compared with the previous one $Existing$.

5. Related work

In Mole [4], it is assumed that an agent never migrates while communicating with another. Thus, if corresponding agents move, their communication is implicitly terminated. It maintains a trail of pointers

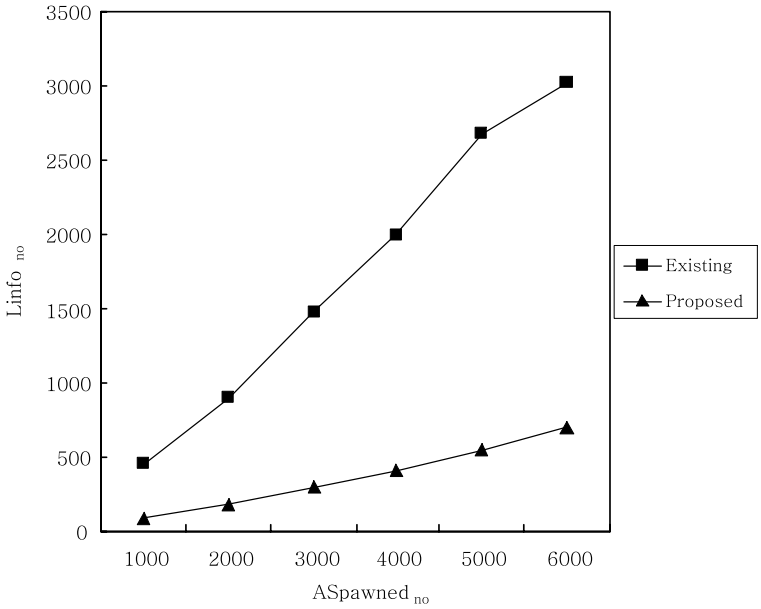

Fig. 12. $LInfo_{no}$ vs. $ASpawned_{no}$.

of each agent for faster communication. But, this is exploited only in the context of a orphan detection protocol. Also, Mole supports group communication for a set of agents being non-mobile during a set of information exchanges.

A broadcast-based mobile agent communication mechanism was proposed by Murphy and Picco [18]. The mechanism guarantees transparent and reliable inter-agent communication and can also provide multicast communication for a set of agents. But, to locate the message destination, it has to contact every visiting host in the network. Thus, its large traffic overhead makes broadcasts impractical in large-scale mobile agent systems.

Concordia [27] supports inter-agent communication implemented via events. Each agent cannot raise and listen to events until it should register with an event manager. Subsequently, the sender agent connects with the event manager and posts an event with its request. As soon as the event manager receives a particular event, it informs all registered agents of the event. Although an agent migrates to another node, it can still receive requests from the event manager. It can register with any number of event managers. But, for one agent to communicate with another agent, the recipient agent should know the event manager location exactly.

Belle [6] proposed a hierarchical structure-based mechanism to form a location directory consisting of a hierarchy of servers. The location server at each level keeps lower-level object location information. For each object, the information is either a pointer to an entry at a lower-level location server or the agent's actual current location. However, this hierarchy cannot always be easily formed, especially in the Internet environment. Moreover, this mechanism may cause useless hops to be taken along the hierarchy.

In [24], a resending-based message delivery mechanism was introduced to use TCP-like sliding-window protocols for detecting the loss of a message and retransmitting it. After several resendings, the sender contacts the location server and transmits the message to the new destination. But, the mechanism needs synchronizing the message routing with agent migration.

A distributed location directory management mechanism for mobile hosts [23] was proposed to be able to adapt to changes in geographical distribution of mobile hosts population in the network and to changes

in mobile host location query rate. The mechanism replicates location information about mobile hosts at $O(\sqrt{m})$(m is the total number of base stations in the system). This mechanism allows frequently queried mobile hosts to have their location information saved on a greater number of base stations. Thus, it may improve lookup and update performance by using replication. However, it is unsuitable for mobile agent communication because the highly dynamic nature and extremely fast migration speed of software mobile agents unlike mobile hosts make it very difficult to get the information about the distribution patterns and location query rates of a large number of mobile agents immediately, which may result in enormously high communication overhead.

Moreau [16] proposed a distributed directory service and message routing mechanism based on forwarding pointers for mobile agents. In this mechanism, each mobile agent is associated with a timestamp that is increased every time the agent migrates. This timestamp avoids cyclic routing when the agent migrates to previously visited hosts. Also, this mechanism introduces a type of message $inform$, containing an agent's location and the timestamp it had at that location, in order to reduce chains of forwarding pointers. A host that receives an inform message is allowed to update its table if the received information is more recent than the one it had. However, it specifies no concrete and efficient solutions for this purpose, for example, when inform messages must be sent, and which host they should be sent to.

Feng [8] introduced a mailbox-based mechanism to provide location-independent reliable message delivery. It allows messages to be forwarded at most once before they are delivered to their receiving agents. Also, the movement of agents can be separated from that of their mailboxes by determining autonomously whether each mailbox is migrated to its owner agent. However, uncertainty of message delivery to mailboxes may result in useless early pollings. On the other hand, even if urgent messages are forwarded to a mailbox on time, they can be delivered to its corresponding agent very late depending on the agent's polling time. Moreover, whenever each mailbox moves, its new location information should be broadcasted to every node where the mailbox has visited. This may incur high traffic overhead if assuming most agents are highly mobile.

Coordination models offer a more asynchronous form of communication, typically involving a tuple space [7]. As coordination spaces are non-mobile, they may result in the centralization problem like the home agent based mechanism. To alleviate this problem, the tuple space can be implemented in a distributed manner. But, consistency maintenance overhead is not negligible. Moreover, in this approach, coordinated processes have to poll tuple spaces, which leads to some difficulties with respect to both communication and computation. As a result, tuple spaces generally provide a mechanism by which registered clients can be informed of the arrival of a new tuple. If clients are mobile, the problem of how to deliver such notifications correctly occurs. When the tuple space is mobile [20], message delivery to the space should be also considered.

6. Conclusion

This paper introduced a scalable distributed directory and message delivery mechanism for mobile agents to require a small size of storage of each service node for agent location management and incur low message forwarding overhead while reducing the dependency on the home node. To achieve this goal, this mechanism forces only a small number of nodes among its visiting ones of each agent to become forwarders. It has each agent register its location with its current locator on every migration until it arrives at the next locator of the agent. This behavior may result in slightly higher location update cost per agent migration compared with that of the previous forwarding pointer-based mechanism. However,

if each agent determines some among its visiting nodes as forwarders by properly considering several performance factors, the gap between the two costs may be almost negligible. Moreover, its message delivery algorithm enables each sending agent to communicate with mobile agents very fast by effectively managing and using their bindings in its agent location cache. Our experimental results demonstrated that the proposed mechanism considerably reduces both the amount of agent location information maintained by each service node and the delivery time of each message destined to a mobile agent compared with the existing one.

For future work, our research group are currently developing some methods for selecting appropriate forwarders of each mobile agent according to the changes which the agent senses in its environment related to location updating and message delivery costs, security policies, network latency and topology, communication patterns. In addition, we attempt to extend our mechanism to address fault-tolerance of forwarders of each mobile agent by using some effective information redundancy technique while reasonably preserving scalability of our mechanism.

References

[1] K. Alexandros and S. Joseph, A Trustworthy Mobile Agent Infrastructure for Network Management, *In Proc. of the 10th IFIP/IEEE International Symposium on Integrated Network Management* (2007), 383–390.
[2] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin and H. Y. Song, Parsec: A Parallel Simulation Environments for Complex Systems, *IEEE Computer* (1998), 77–85.
[3] F. Baschieri, P. Bellavista and A. Corradi, Mobile Agents for Qos Tailoring, Control and Adaptation over the Internet: The UbiQoS Video on Demand Serbvice, *In Proc. of the 2nd International Symposium on Applications and the Internet* (2002), 109–118.
[4] J. Baumann, F. Hohl, K. Rothermel and M. Strasser, Mole-Concepts of a Mobile Agent System, *World Wide Web* **1**(3) (1998), 123–137.
[5] P. Bellavista, A. Corradi and C. Stefanelli, The Ubiquitous Provisioning of Internet Services to Portable Devices, *IEEE Pervasive Computing* **1**(3) (2002), 81–87.
[6] W. Belle, K. Verelst and T. D'Hondt, Location transparent routing in mobile agent systems merging name lookups with routing, *In Proc. of the 7th IEEE Workshop on Future Trends of Distributed Computing Systems* (1999), 207–212.
[7] G. Cabri, L. Leonardi and F. Zambonelli, Reactive Tuple Spaces for Mobile Agent Coordination, *In Proc. of the 2nd International Workshop on Mobile Agents*, Lecture Notes In Computer Science **1447** (1998), 237–248.
[8] J. Cao, X. Feng, J. Lu and S. Das, Mailbox-based scheme for mobile agent communications, *IEEE Computer* **35**(9) (2002), 54–60.
[9] J. Cui and H. Chae, Mobile Agent based Load Balancing for RFID Middlewares, *In Proc. of International Conference on Advanced Computer Technology* (2007), 973–978.
[10] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. Murphy and G. Picco, Mobile Data Collection in Sensor Networks: The TinyLime Middleware, *Journal of Pervasive and Mobile Computing* **4**(1) (2005), 446–469.
[11] J. Desbiens, M. Lavoie and F. Renaud, Communication and tracking infrastructure of a mobile agent system, *In Proc. of the 31st Hawaii International Conference on System Sciences* **7** (1998), 54–63.
[12] A. Fuggetta, G.P. Picco and G. Vigna, Understanding Code Mobility, *IEEE Transactions on Software Engineering* **24**(5) (1998), 342–361.
[13] M. Fukuda, K. Kashiwagi and S. Kobayashi, AgentTeamwork: Coordinating Grid-Computing Jobs with Mobile Agents, *In Special Issue on Agent-Based Grid Computing, International Journal of Applied Intelligence* (2006).
[14] D. Lange and M. Oshima, *Programming and Deploying Mobile Agents with Aglets*, Addison-Wesley (1998).
[15] Z. Li and M. Parashar, A Decentralized Agent Framework for Dynamic Composition and Coordination for Autonomic Applications, *In Proc. of the 3rd International Workshop on Self-Adaptive and Autonomic Computing Systems*, Copenhagen, Denmark (2005), 165–169.
[16] L. Moreau, Distributed Directory Service and Message Router for Mobile Agents, *Science of Computer Programming* **39**(2–3) (2001), 249–272.
[17] L. Moreau and D. Ribbens, Mobile Objects in Java, *Scientific Programming* **10**(1) (2002), 91–100.
[18] A.L. Murphy and G.P. Picco, Reliable Communication for Highly Mobile Agents, *Journal of Autonomous Agents and Multi-Agent Systems* **5**(1) (2002), 81–100.
[19] A.L. Murphy and G.P. Picco, Using Lime to Support Replication for Availability in Mobile Ad Hoc Networks, *In Proc. of the 8th International Conference on Coordination Models and Languages* (2006), 194–211.

[20] A.L. Murphy, G.P. Picco and G.-C. Roman, Lime: A Coordination Middleware Supporting Mobility of Hosts and Agents, *ACM Transactions on Software Engineering and Methodology* **15**(3) (2006), 279–328.
[21] C. Perkins, IP Mobility Support, *RFC 2002* (1996).
[22] V. Pham and A. Karmouch, Mobile Software Agents: An Overview, *IEEE Communications Magazine* **36** (1998), 26–37.
[23] R. Prakash and M. Singhal, A Dynamic Approach to Location Management in Mobile Computing Systems, *In Proc. of the International Conference on Software Engineering and Knowledge Engineering* (1996), 488–495.
[24] M. Ranganathan, M. Bednarek and D. Montgomery, A Reliable Message Delivery Protocol for Mobile Agents, *In Proc. of the 2nd International Symposium on Agent Systems and Applications and the 4th International Symposium on Mobile Agents*, Lecture Notes In Computer Science **1882** (2000), 206–220.
[25] K. Rothermel and M. Schwehm, Mobile Agents, *Encyclopedia for Computer Science and Technology* **40** (1999), 155–176.
[26] R. Tahboub and V. Lazarescu, Novel Approach for Remote Energy Meter Reading Using Mobile Agents, *In Proc. of the 3rd International Conference on Information Technology* (2006), 84–89.
[27] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young and B. Peet, Concordia: An Infrastructure for Collaborating Mobile Agents, *Lecture Notes In Computer Science* **1219** (1997).
[28] J. Zhang, A mobile agents-based approach to test the reliability of web services, *International Journal of Web and Grid Services* **2**(1) (2006), 92–117.

Jinho Ahn received his B.S., M.S. and Ph.D. degrees in Computer Science and Engineering from Korea University, Korea, in 1997, 1999 and 2003, respectively. He has been an assistant professor in Department of Computer Science, Kyonggi University. His research interests include distributed computing, fault-tolerance, sensor networks and mobile agent systems. He has published more than 50 papers in refereed journals and conference proceedings and served as program committee member or session chair in several domestic/international conferences.