

---

## Mobile agent location management in global networks

---

R.B. Patel\*

Department of Computer Engineering,  
M.M. Engineering College,  
Mullana 133203, Haryana, India  
E-mail: patel\_r\_b@indiatimes.com  
\*Corresponding author

Nikos E. Mastorakis

Department of Computer Science,  
Military Institute of University Education/Hellenic Naval Academy,  
Terma Hatzikyriakou 18539, Piraeus, Greece  
E-mail: mastor@wseas.org

Kumkum Garg

Department of Electronics and Computer Engineering,  
IIT Roorkee,  
Roorkee 247667, Uttaranchal, India  
E-mail: kgargfec@iitr.ernet.in

**Abstract:** Mobility management is a necessity in highly dynamic and large-scale Mobile Agents (MAs) networks, especially in a multi-region environment in order to control and communicate with agents after launching. Existing mechanisms for locating MAs are not efficient as these do not consider the effect of location updates on migration time and produce network overload. This paper presents a hierarchical model for location management of MAs in global networks. Three protocols are developed, namely search, update and search-update. The location management technique uses one combination of search, update and search-update protocols throughout execution. Three cases are considered for Update and Search-Update Protocols. Thus, nine combinations of location management protocols are generated, from which an agent can dynamically select one as per requirement, to communicate with other agents on the global network. We have implemented these protocols on the system developed at IIT Roorkee, to evaluate the performance. Results indicate that overhead generated by these protocols does not affect the actual agent response and migration time.

**Keywords:** location management; Mobile Agent; MA; update protocol; search protocol and search-update protocol.

**Reference** to this paper should be made as follows: Patel, R.B., Mastorakis, N.E. and Garg, K. (2007) 'Mobile agent location management in global networks', *Int. J. Information and Communication Technology*, Vol. 1, No. 1, pp.26–37.

**Biographical notes:** R.B. Patel received PhD from IIT Roorkee in Computer Science and Engineering, PDF from Hiest, Athens, Greece, an MS from BITS Pilani and a BE in Computer Engineering from M.M.M. Engineering College, Gorakhpur, UP. He has published about 50 research papers in International/National Journals and Refereed International Conferences. He has been awarded for the Best Research paper by Technology Transfer, Colorado, Springs, USA, for his security concept provided for mobile agents on open network in 2003. He has written five books for engineering courses. His current research interests are in Mobile and Distributed Computing, Mobile Agent Security and Fault Tolerance, development infrastructure for mobile and peer-to-peer computing, Device and Computation Management, Cluster Computing, etc.

Nikos E. Mastorakis received a PhD in Electrical Engineering and Computer Science from the National Technical University of Athens. He is a Full time Professor and the Head of the Department of Computer Science at the Military Institutions of University Education -Hellenic Naval Academy, Greece since 1994. He was *Editor-in-Chief* in more than 12 Journals. He has published more than 200 research papers in International Journals, transaction and Conferences. His research areas are Mobile Agents, Neural Networks, Genetics Algorithms, E-Commerce, Fuzzy Systems, etc.

Kumkum Garg has been on the faculty of the Department of Electronics and Computer Engineering, IIT Roorkee, since 1976, having done her BE and ME in 1971 and 1976, respectively, from the same Department. She received her PhD in Distributed Computer Systems from Imperial College of Science and Technology, University of London, UK in 1984. Currently she is coordinating a natural programme for HRD development in IT at IIT, Roorkee. Her research interests are in Computer Networks, Natural Language Processing, Evolutionary Computing, Distributed Computing, Mobile Agent and Mobile Agent Security. She is a senior member of the IEEE Computer Society and fellow of IE (I).

---

## 1 Introduction

A Mobile Agent (MA) (Picco, 2001) is a software process, which can move autonomously from one physical network location to another. The agent performs its job wherever and whenever it is found appropriate and is not restricted to be colocated with its client. Thus, there is an inherent sense of autonomy in the mobility and execution of the agent. Agents can be seen as automated errand boys who work for users. MA research evolved over the past years from the creation of many different monolithic Mobile Agent Systems (MASs), often with similar characteristics and built by research groups spread all over the world, for optimisation and better understanding of specific agent issues (Picco, 2001; Tripathi et al., 2001).

As large-scale MASs are the next trend following the popularity of MA technology, the collaborations between roaming agents has increased. There should be an efficient mobility management for locating MAs as part of the agent communication platform. The basic operations associated with mobility management are:

- 1 A roaming agent updates its location frequently to the central management server, that is, a directory server (Stanski et al., 1998).

- 2 The agent management server refreshes the current location record of the agent in its location database.
- 3 When there is a request asking for the location of the agent, the management server searches the database and replies with the current location of the MA. Besides these three basic steps, the server may also process issues such as out-of-date location records. Most existing MASs have provided partial mobility management, by defining different naming and locating mechanisms.

The ability to locate MAs while they are migrating from one node to another one is of great importance for the development of agent-based applications which have to work in geographically distributed environments (van Steen et al., 1998). This issue becomes even more important when focus is shifted from distributed application limited in space, to distributed application whose environment is spread all over the Internet. None of the Java-based MA platform provides a comprehensive, effective location management system. In any case these mechanisms are strictly tied with the platform that they are designed for without exploiting existing techniques for searching or locating objects in the Internet. When a global environment such as the Internet is considered, a centralised naming protocol quickly becomes a bottleneck for the system, providing poor performance. Distributed techniques and algorithms are often more effective even if their implementation is more difficult (Bernardo and Pinto, 1998; Lazar et al., 1998; Roth and Peters, 2001).

Location management is an important issue in MA computing. It consists of location updates, searches and search-updates: An update occurs when a MA changes location. A search occurs when a MA/Agent Host (AH) (Patel and Garg, 2004; Patel, 2004) wants to communicate with a MA whose location is unknown to the requesting agent/host. A search-update occurs after a successful search, when the requesting agent/host updates the location information corresponding to the searched MA. The goal of a good location management protocol should be to provide efficient searches and updates. The number of messages sent, size of messages and distance the messages need to travel, characterise the cost of a location search and update protocol. An efficient location management protocol should attempt to minimise all these quantities. Hence, a new protocol is required that would generate minimum overhead and be suitable for both global and local area networks.

This paper reports several location management protocols based on a hierarchical tree structure database. It also reports on the results of implementations carried out to evaluate the performance of proposed location management protocols for various call and mobility patterns. Platform for Mobile Agent Distribution and Execution (PMADE), developed at IIT Roorkee, is used as the development platform (Patel and Garg, 2004; Patel, 2004).

The rest of this paper is organised as follows: Section 2 reports on a system model for a distributed system with MAs. Section 3 presents proposed location management protocols. Evaluation results are shown in Section 4. Conclusions are given in Section 5.

## **2 System model**

In PMADE, agent location is based on some assumptions for the distributed environment, as shown in Figure 1. We have assumed that the global network environment is divided into network domains, regions (subnetworks) and AHs



fixed hosts. A host that can move while retaining its network connection is called a MH. The static network comprises of the fixed hosts and the communication links between them. Some of the fixed hosts, called Base Host (BH) are augmented with a wireless interface and they provide a gateway for communication between the wireless network and the static network (Patel, 2004).

Due to the limited range of wireless transceivers, a MH/MA can communicate with a BH. The geographical area covered by a region is a function of the medium used for wireless communication. Currently, the average size of a region is of the order of 1–2 miles in diameter. As the demand for services increase, the number of regions may become insufficient to provide the required grade of service. Region splitting can then be used to increase the traffic handled in an area without increasing the bandwidth of the system. A MA communicates with one BH at any given time. The BH is responsible for forwarding data between the MH/MA and the static network.

Due to mobility, MH/MA may cross the boundary between two regions while being active. Thus, the task of forwarding data between the static network and the MH/MA must be transferred to the new regions. This process, known as handoff, is transparent to the mobile user (Kessler et al., 1995). The initiative for a handoff can come from the MH or the BHs. Handoff helps to maintain an end-to-end connectivity in the dynamically reconfigured network topology.

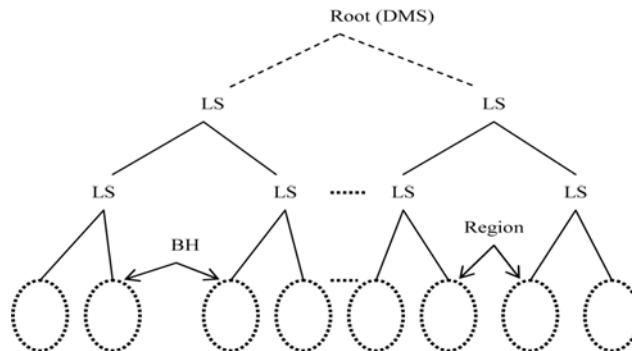
### 3 Location management protocol

A location management protocol is a combination of a search protocol, an update protocol and a search-update protocol. Only the location management protocols in the absence of a Home Location Server (HLS) are discussed in this paper.

#### 3.1 Logical network architecture

A Global network consists of MAs, MASs and Location Servers (LSs). The Logical Network Architecture (LNA) is a hierarchical structure (a tree with  $H$  levels) consisting of BHs and LSs. As shown in Figure 2, the BHs are located at the leaf level of the tree. Each BH maintains information about the agents residing in its region. The other nodes in the tree are called LSs. Each LS maintains information regarding MAs residing in its subtree.

**Figure 2** Logical network architecture



Each communication link has a weight attached to it. The weight of a link is the cost of transmitting a message on the link. Let  $l[\text{src}][\text{dest}]$  represent the link between nodes  $\text{src}$  and  $\text{dest}$  and let  $w(l)$  represent the weight (or cost) of link  $l$ . The cost depends on the size of the message, the distance between the hosts (agents), and the bandwidth of the link. For analysis purposes, we assume that, for all  $l$ ,  $w(l) = 1$ . Essentially, the cost metric is the number of messages sent.

### 3.2 Data structures

There is a unique ‘home’ address for every MA. The home address is the identifier/name of the MA. The ‘physical’ address of a MA might change, but its home address remains the same, irrespective of the agent location (Patel, 2004; Stefano and Santoro, 2002). Each LS maintains an address matching table that maps the home address to the physical address of the MAs residing in the subtree beneath it. Thus, the problem of location management basically focuses on the management of the address matching tables.

There is a location entry in a LS corresponding to an agent  $A$ , if it is in a region in the subtree under LS. If  $A$  moves to a region which is not in the subtree under LS, then the entry corresponding to  $A$  is updated at LS. All the nodes maintain location information using three-tuples which have the following elements:

- 1 MA identifier (id) (given by agent naming server)
- 2 forwarding pointer destination ( $fp\_dest$ ) and
- 3 time at which last forwarding pointer update took place ( $fp\_time$ ).

Each LS maintains a three-tuple for each MA residing in the subtree beneath it and each BH maintains a three-tuple for each MA residing in its region. The default value of  $fp\_dest$  and  $fp\_time$  is NULL. If the  $fp\_dest$  field of an agent  $A$  is NULL in LS  $L$ , then,  $A$  is not in a region in the subtree under  $L$ . Let us suppose that we are using a protocol which uses forwarding pointers for location updates. Let  $A$  reside initially in the region  $r$ . The BH of region  $r$  will have an entry  $(A, \text{NULL}, \text{NULL})$ . Let there be a LS  $L$  which maintains information about the agents residing in  $r$ . There will be an entry  $(A, r, \text{NULL})$  corresponding to  $A$  at  $L$ . Let  $A$  move to a new region  $r'$ , which is not a part of the subtree of  $L$ . Let  $t$  be the local time at the BH of  $r$  when change of location of  $A$  is recorded at BH. Let  $t'$  be the local time at  $L$  when the change of location of  $A$  is recorded at  $L$ . Thus, the location information of  $A$  will be  $(A, r', t')$  at  $L$  and  $(A, r', t)$  at BH of region  $r$ .

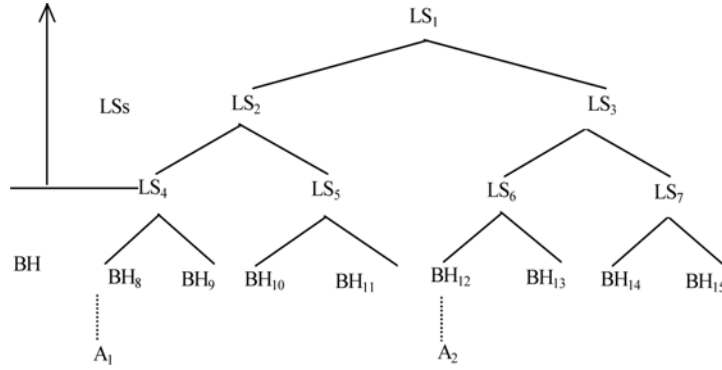
*Note:* The above data structures contain  $fp\_time$  field to store time. The  $fp\_time$  entry for a data structure on a node, say  $v$ , contains the local time at node  $v$  when the data structure was last modified. We will denote this time by  $t$  in the following. It should be noted that the correctness of the algorithms does not require the clocks at various nodes to be tightly synchronised.

### 3.3 Initial conditions

It is assumed that, initially location information of the MAs is stored in the corresponding LSs, that is, each LS has the correct location information for all the agents residing in the region in its subtree. Thus, the root LS should have the correct location information of all agents in the system. In Figure 3, nodes  $LS_{1-7}$  are LSs and

$BH_{8-15}$  are BHs. There are two MAs  $A_1$  and  $A_2$ . In the initial state, agent  $A_1$  is in region 8 and  $A_2$  is in region 12. Initially, the correct location information of agent  $A_1$  will be available at LSs  $LS_4$ ,  $LS_2$  and  $LS_1$ . Likewise, the location information of  $A_2$  will be available at LSs  $LS_6$ ,  $LS_3$  and  $LS_1$ . Thus, the location information of an agent is available at all the LSs located on the path from its current BH to the root.

**Figure 3** Example of proposed network architecture



### 3.4 Update protocol

The protocol for updating the location information at the LSs and the BHs, when a MA moves, is as follows: Let  $src$  and  $dest$  be the source and destination regions, respectively. Let  $A$  be the identifier of the MA. Let  $t$  denote the local time at a node when a change in location of  $A$  is recorded at that node. The value of  $t$  will be different at different nodes. For this protocol we have considered three cases as follows.

*Case 1 Single Updates (SU):* in this the update takes place only at the BH of the source and destination regions. A forwarding pointer is kept at the source BH. The updated entry at the source BH becomes  $(A, dest, t)$ . An entry for agent  $A$ ,  $(A, NULL, NULL)$  is added at the destination BH. The location information at the LSs are not updated. The cost of update is zero because there is no update message being sent.

*Case 2 Full Updates (FU):* upon a move, apart from BHs involved (i.e., BH of the source and destination regions), location updates take place in all the LSs located on the path from the BH of the source and destination region to the root. Details are as follows:

*Source region:* (1) *At the BH:* for agent  $A$ , set  $fp\_dest = dest$  and  $fp\_time = t$ . The updated entry for agent  $A$  at the BH becomes  $(A, dest, t)$ . (2) *All LSs on the path from src to the root:* the BH of  $src$  sends update message to these LSs. Upon receipt of the update message, the LSs update the entry for  $A$  to  $(A, dest, t)$ .

*Destination region:* (1) *At the BH:* an entry  $(A, NULL, NULL)$  is added for agent  $A$ . If there was an old entry for  $A$ , it is overwritten by this new entry. At any node, there can be only one entry per agent. (2) *All LSs on the path from dest to the root:* the BH of  $dest$  sends update message to these LSs. Upon receipt of the update message the LSs create an entry. If there was an old entry, it is overwritten by this new entry.

Therefore, in an  $H$ -level tree, the update cost (the cost metric is number of messages) per move is  $2(H-1)$ . Suppose in Figure 3, agent  $A$  moves from  $BH_8$  to  $BH_{14}$ . A forwarding pointer to  $BH_{14}$  will be kept at  $BH_8$ .  $BH_8$  sends update messages to LSs  $LS_4$ ,  $LS_2$  and  $LS_1$  and these LSs also create a forwarding pointer to  $BH_{14}$ . An entry for  $A_1$  will be made at  $BH_{14}$ .  $BH_{14}$  sends update message to LSs  $LS_7$ ,  $LS_3$  and  $LS_1$  and these LSs also make an entry for agent  $A_1$ .

*Case 3 Limited Update (LU):* update in the location information takes place at a limited number of levels of LSs in the tree. Here, updates occur at  $m(<H)$  lower levels of LSs on the path to the root. Updates at these LSs are similar to the FU. The LSs at levels higher than  $m$  are not updated. Thus, the update cost per move is  $2m$ . Let the value of  $m$  be chosen to be 1. Suppose in Figure 3, agent  $A_1$  moves from  $BH_8$  to  $BH_{14}$ . The forwarding pointer to  $BH_{14}$  will be kept at  $BH_8$ .  $BH_8$  sends an update message to  $LS_4$  and  $LS_4$  maintains forwarding pointer to  $BH_{14}$ . An entry for  $A_1$  will be made at  $BH_{14}$ .  $BH_{14}$  sends an update message to  $LS_7$ , who makes an entry for agent  $A_1$ .

### 3.5 Search protocol

If agent  $A$  in region  $R$  wants to communicate with another agent  $A'$ ,  $A$  has to know the location of  $A'$ . This requires that agent  $A$  search for agent  $A'$ . As stated earlier, we do not make explicit use of HLSs for searches. The search process in the absence of a HLS is as follows.

- 1 If the BH of  $R$  has no location information for  $A'$ , it forwards the location query to the next higher-level LS on the path to the root.
- 2 If the LS does not have any location information for  $A'$ , it again forwards the location query to the next higher-level LS on the path to the root.
- 3 Repeat 1 & 2 until a LS which has location information for  $A'$  is reached.
- 4 If the location information (i.e., region identifier, say  $S$ ) for  $A'$  is obtained, the location query is forwarded to the BH of region  $S$ . Agent  $A'$  will either be in region  $S$  or the BH will have a forwarding pointer corresponding to  $A'$ .
- 5 If  $A'$  is in region  $S$ , the search is complete. Else, a chain of forwarding pointers is traversed until BH of the containing agent  $A'$  is reached.

### 3.6 Search-update protocol

Location management becomes more efficient if the location updates also take place after a successful search. For example, suppose there is an agent  $A$  that frequently calls  $A'$ . It may be useful to update the location information of  $A'$  after a successful search, so that if  $A$  calls again, the search cost is likely to be small. The location information update takes place at the BH of the caller agent. Let agent  $A$  be the caller at the source and  $A'$  be the agent to be searched at the destination host. Let the location of  $A$  and  $A'$  be  $K$  and  $K'$ , respectively. The following are the three cases to update location information upon a search.

*Case 1 No Update (NU):* there are no location updates, the *fp\_time* field of the entry corresponding to  $A'$  at the BH on the search path is updated to the current time at the BH. The cost is zero. This is because the update of the time field could be done during the search process itself and no additional message needs to be sent for this purpose. The update in *fp\_time* is done to avoid purging of the forwarding pointer data at the BHs. The purge protocol is explained in the next section.

*Case 2 Jump Update (JU):* a location update takes place only at the caller agent's BH, that is, BH of region  $K$ . The entry for  $A'$  at the BH of region  $K$  is set to  $(A', K', t)$ , where  $t$  is the local time at the BH when the location information is updated. This update cost is 1. This is because only one message needs to be sent from BH of  $K'$  notifying the location information of agent  $A'$ .

*Case 3 Path Compression Update (PCU):* upon a successful search, a location update takes place at all the nodes in the search path. All the LSs on the search path have the entry of  $A'$  updated to  $(A', K', t)$  where  $t$  is the local time at the LS when the location information is updated. All the BHs on the search path including the caller agent's BH have an entry of  $A'$  updated to  $(A', K', t)$ , where  $t$  is the local time at the BH when the location information is updated. In Figure 3, let agent  $A_1$  calls agent  $A_2$ . Suppose the location information of  $A_2$  is available only at the  $LS_6$ ,  $LS_3$  and  $LS_1$ . Using the search protocol described previously, the search path will be  $BH_8 \rightarrow LS_4 \rightarrow LS_2 \rightarrow LS_1 \rightarrow BH_{12}$ . The location updates take place at  $LS_4$ ,  $LS_2$  and  $LS_1$  and  $BH_8$ . The update cost is the length of the search path that is in this example is 4.

### 3.7 Purging protocol

We need to periodically purge stale forwarding pointers at the LSs and the BH. This should be done in order to

- 1 save storage space at the nodes and
- 2 avoid storing stale location information.

We use a parameter called Maximum Threshold Call Interval (MTCI) to decide whether to purge the forwarding pointer information or not. Let the current time be *curr\_time*. If *fp\_time*  $\neq$  NULL and  $\text{curr\_time} - \text{fp\_time} \geq \text{MTCI}$  then the entry for the agent is purged from the  $BH_i$ , if some other agents in the system have recently used the forwarding pointer information of  $BH_i$ . In the LSs, if  $\text{curr\_time} - \text{fp\_time} \geq \text{MTCI}$  for MA, the location entry for that agent is purged.

When SU and LU cases of update protocol are used, the forwarding pointers at higher-level LSs do not get updated and become stale. Thus, these forwarding pointers get purged periodically. However, some of the searches for the agent might reach the higher levels. If the LSs at the higher levels do not have information about the agent, the root has to broadcast to determine the location. To avoid this, the forwarding pointers at the LSs on the path to the root from the current BH must be updated periodically along with purging. The current BH of each MA achieves this by sending a location update message to the LSs on the path to the root.

*Note:* the *fp\_time* value for an agent residing in the region will be NULL. So we are considering agents which are currently not residing in the BH's region and whose forwarding pointer information is stored at the BH.

#### 4 Evaluation results

Our tests took place in a 10/100 MBps switched LAN that connects 850 workstations and personal computers and is used by about 500 hundred researchers and students. We ran PMADE equipped with the developed protocols on several P-4, 3 GHz machines. The Agent Submitter (AS) node and AH nodes have 256 MB main memory, while the LS (AH at the root) has 512 MB. We used the j2sdk 1.4.1 Java Virtual Machine with native thread support.

First, we tested the capacity and performance of our storage backend. The LS (root AH) was able to hold up to  $4 \times 10^6$  entries before the system ran out of memory (Figure 2). This means that, given an extreme of  $8 \times 10^8$  Internet users (*NUA estimates there were more than 605.60 million users online in the Internet on September 2002*, [http://www.nua.com/surveys/how\\_many\\_online](http://www.nua.com/surveys/how_many_online)) each running 100 MAs simultaneously, about 20,000 LSs would be required to keep all entries. This is less than 0.0057% of the hosts in the Internet, according to ISC estimates (*ISC estimates there were more than 350,000,000 hosts in the Internet in January 2005*, <http://www.isc.org/ds>) at the time of writing.

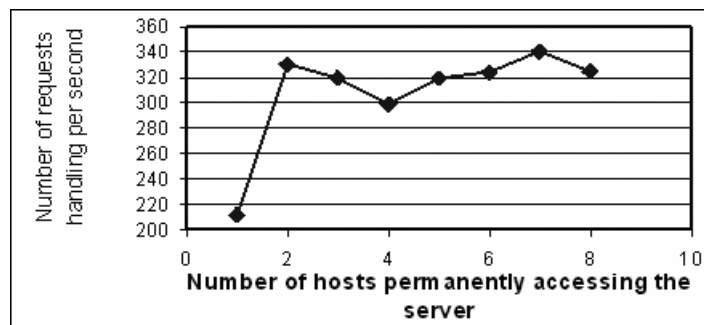
Next, we let up to eight agents/ASs send requests concurrently. Table 1 gives the response rates we measured in tests with a single agent/AS, sorted by request type. Secured registration was slowest, as could be expected. However, this type of request is required only once per agent. In this test the LS handled about 400 agent lookup requests per second, which includes processing overhead at the AS (ASs start requests in parallel threads). Figures 4 and 5 show the response rates we measured for concurrent lookup requests with one to eight agents/ASs (average of 6000 measured values taken).

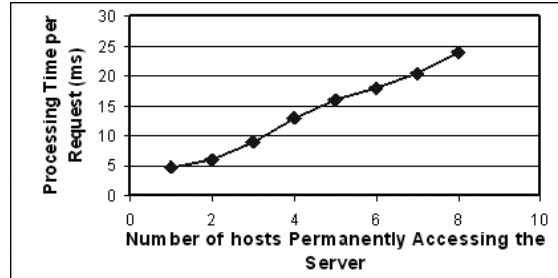
**Table 1** Size of request packets and average processing time of the searching service with one agent/AS, by request type

Type	Length	Mean time	Requests/sec	Action of
Lookup	32 bytes	4.7 ms	313	Location search
Registration secured	431 bytes <sup>a</sup>	11 ms	15	Init
Update	103 bytes <sup>a</sup>	1 ms	150	Location update
Register unsecured	103 bytes <sup>a</sup>	5 ms	270	LS

<sup>a</sup>The lengths marked which might differ depending on the length of the stored location reference

**Figure 4** The average number of requests handled by a LS



**Figure 5** Response times

With two or more agents/ASs, the response rate jumps from about 210 requests per second to roughly 332 and remains more or less stable at this mark (with one agent/AS, the AH has idle time, with two or more it becomes congested). Table 2 gives how response times develop with an increasing number of agents/ASs. With about 2626 agents/ASs, requests take longer than 13 sec to process, which causes network connections to time out for few agents.

**Table 2** Agent response time (includes Agent Migration Time, Agent Decryption Time, User/Agent Authentication Time, Result Encryption and Packaging Time)

No. of AH	1	2	4	8	16	32	64	128	256	512
No. of agents/AS	1	2	4	8	16	32	64	128	256	512
Response time	301 ms	572 ms	1 sec, 500 ms	2 sec, 3 ms	4 sec, 3 ms	6 sec, 6 ms	10 sec, 510 ms	21 sec, 21 ms	53 sec	154 sec, 100 ms

We also measured the impact of the location service (search and update) integration on the migration time of MAs in the PMADE. Without location service integration, we measured an average of 140 milliseconds per migration of a simple benchmark agent, compared to with location service (search and update), which we consider tolerable (Table 3).

**Table 3** Effect of LSs on Agent Migration Time (size of agent is considered 10.203 KB)

No. of LSs	1	2	3	4
Agent migration time when LSs are active (ms)	145.7	147	148.1	150
Agent migration time when LSs are not active (ms)	140	140	140	140

## 5 Conclusion

In this paper, we have presented several location management protocols based on a hierarchical tree structure database. These location management protocols use one combination of search, update and search-update protocols throughout the execution.

We have applied these protocols in the real life application implementation developed on PMADE. It is found that overhead generated by them does not affect the actual agent response and migration times.

## References

- Kessler, I. Bar-Noy, and Sidi, M. (1995) 'Mobile users: to update or not to update?' *ACM-Baltzer Journal of Wireless Networks*, Vol. 1, No. 2, pp.175–186.
- Bernardo, L. and Pinto, P. (1998) 'A scalable location service with fast update responses', *IEEE Global Telecommunications Conference (GLOBECOM '98)*, 8–12 November, Sydney, Australia, pp.2876–2881.
- Lazar, S., Whereon, I.P. and Sidhu, D.P. (1998) 'A scalable location tracking and message delivery scheme for mobile agents', *Proceedings of the Seventh Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'98)*, 17–19 June, Palo Alto, CA: IEEE Computer Society, pp.243–249.
- Patel, R.B. (2004) 'Design and implementation of a secure mobile agent platform for distributed computing', PhD Thesis, Department of Electronics and Computer Engineering, IIT Roorkee, India, August.
- Patel, R.B. and Garg, K. (2004) 'A new paradigm for mobile agent computing', *WSEAS Transaction on Computers*, Vol. 3, No. 1, pp.57–64.
- Picco, G.P. (2001) 'Mobile agents: an introduction', *Microprocessors and Microsystems*, Vol. 25, No. 2, pp.65–74.
- Roth, V. and Peters, J. (2001) 'A scalable and secure global tracking service for mobile agents', G. Picco (Ed). *Proceedings of the Fifth International Conference on Mobile Agents (MA2001)*, Atlanta, Georgia, USA, LNCS 2240, Springer Verlag, pp.169–181.
- Stanski, P., Thompson, D., Nzama, M., Zaslavsky, A. and Craske, N. (1998) 'Automating directory services for mobile agent tracking', *IEEE Global Telecommunications Conference (GLOBECOM '98)*, 8–12 November, Sydney, Australia, pp.1947–1951.
- Stefano, D. and Santoro, C. (2002) 'Locating mobile agents in a wide distributed environment', *IEEE Transaction on Parallel and Distributed Systems*, Vol. 13, No. 8, pp.844–864.
- Terry, D.B. (1985) 'Distributed name servers: naming and caching in large distributed computing environments', PhD Thesis, University of California, Berkely, Available as UCB/CSD Technical Report 85-228 and as Xerox PARC Technical Report CSL-85-1.
- Tripathi, R., Ahmed, T. and Karnik, N.M. (2001) 'Experiences and future challenges in mobile agents programming', *Microprocessors and Microsystems*, Vol. 25, No. 2, pp.121–129.
- van Steen, M., Hauck, F.J., Homburg, P. and Tanenbaum, A.S. (1998) 'Locating objects in wide-area systems', *IEEE Communications Magazine*, January, pp.104–109.