Enhanced Communication Scheme for Mobile Agents

Geetha Priya. B, Suba. S, Tanya Bansal, P.Boominathan
Department of Computer Science and Engineering
Velammal Engineering College
Chennai – 600066, India
geethapriya.rt@gmail.com, subavec@gmail.com, tanyapbansal@gmail.com, bominathan@yahoo.com

***Abstract*— Mobile agent technology is a promising field which has a great scope in the areas of Networking, Distributed Systems and Grid Systems. The frequent movement of the agents poses challenges for the design of an efficient communication protocol for mobile agents. In this paper we propose an improved communication scheme for mobile agents which overcomes the following issues. First, the problem of overloading of the agent is solved by provision for generating a twin agent when required. Also urgent messages are given priority by using a message queue which sorts the messages according to priority frequently. Our scheme also provides a mechanism for the mobile agent to automatically update itself according to the environmental changes in a predictable and visible manner. Also it reduces the resource overhead over the network dynamically by temporarily destroying the idle agents and reconstructing them when needed.**

Keywords-mobile agent; communication scheme; agent overloading; message queue

I. INTRODUCTION

Mobile agents have been widely argued to be an important enabling technology for future distributed systems. Mobile agent technology has great potential for use in networking. In recent years, mobile agent computing has emerged as a new paradigm in developing applications in various areas. Its applications range from telecommunications, e-commerce, and information searching to process coordination, monitoring resource usage and network management.

A. Definition of an Agent (System Perspective)

An agent is a software object that is situated within an execution environment [2]. It possesses the following mandatory properties:

1) Reactive: It senses changes in the environment and acts accordingly to those changes;
2) Autonomous: It has control over its own actions;
3) Goal driven: It is pro-active;
4) Temporally continuous: It is continuously executing;

It may also possess any of the following orthogonal properties:

1) Communicative: It able to communicate with other agents;
2) Mobile : It can travel from one host to another;
3) Learning: It adapts in accordance with previous experience;
4) Believable: It appears believable to the end-user.

Mobile Agent [2]: A mobile agent is not bound to the system where it begins execution. It has the unique ability to transport itself from one system in a network to another. The ability to travel, allows a mobile agent to move to a system that contains an object with which the agent wants to interact, and then to take advantage of being in the same host or network as the object.

Communication protocols are among the most important mechanisms in mobile agent systems. In various situations, mobile agents at different hosts must cooperate with one another by sharing information and making decisions collectively. To ensure effective inter-agent communication, these protocols must track target agent locations and deliver messages reliably. The communication algorithms for location-independent message delivery to migrating agents should support two operations: “migrate” and “deliver”.

1) Migrate - facilitating the movement of an agent to a new site.
2) Deliver - locating a specified agent and delivering a message.

In general, inter-agent communication is carried out by message-swapping in mobile agent systems [10]. But in mobile agent communication environment, this scheme is not effective due to the frequent migration of the receiver agent. The abnormal communication phenomenon caused by the physical position of communication body shift was called mobile communication disable [11]. The message chases the target agent and is not able to reach in the case that the migration frequency of the mobile agent is very high. We called this as message chasing [4]. Furthermore, when a certain agent has many messages to deal with, it is unable to process them in a specified time. This phenomenon is referred to as agent overloading in this paper. If agent overloading occurs, the collaboration agents may not get the coordinated messages, leading to failure of collaboration, which can even cause the entire system to collapse.

978-1-4244-4711-4/09/$25.00 ©2009 IEEE

A practical communication mechanism should make the location of an agent transparent.

Due to the asynchronous nature of message passing and agent migration, it is a challenge to guarantee message delivery to highly mobile agents. The mailbox based mobile agent communication mechanism [3, 4, 8] has implemented location transparency and message delivery. The algorithm is preferable in the cases that mobile agents migrate frequently but communicate rarely. In this scheme, each mobile agent has a mailbox which buffers the messages sent to it. The mailbox is detached from its owner agent in the sense that the agent and its mailbox can reside at different hosts. If an agent will not communicate with others at its target host, it will migrate to the host directly and leave its mailbox at the previously located host. In this way the location-updating overhead is saved and the constraints of agents' mobility are decreased. But there is no reliability of message delivery when the mailbox migrates. It can neither reduce the occupation of network resources, nor update the agent automatically. An improved mailbox-based mobile agent communication algorithm is proposed in this paper. It can overcome message loss and ensure the message delivery reliability, provide an effective solution to overloading problem of the mobile agent and ensure that urgent message gets processed first. This algorithm can also decrease the message waiting time effectively and reduce the occupation of resources over the network dynamically. Furthermore, the mobile agent can be updated automatically in this algorithm.

II. EXISTING SYSTEM

In order to implement the location-independence of agent, the peer agent must be located by some mechanism that maps an agent's unique name onto its current location. In a general way, addressing modes which are usually used include: searching mode, logging mode and registration mode [9]. In searching mode, an agent usually is dispatched to visit all possible hosts or broadcasts a message to all the hosts to search the target agent. The overhead is unaffordable when the network is large. In logging mode, since the agent leaves its migration track on the hosts when it passes by, its current location should be attained via following its trail. If the trail information is lost or if one of the hosts is down, the target agent would no longer be found. With the registration scheme, an agent needs to update its location in a predefined directory server (e.g., its home host) that allows agent to be registered, deregistered or located. The directory server can be either a central node or the agent's home host. The disadvantage of having a central node as directory server is that it may become the bottleneck of the system performance and/or a single point of failure.

The agent's home host follows the idea of Mobile IP [7]. The Mobile IP is the protocol designed for IP packets routing to mobile devices. A mobile host registers its care-of- address with its home host and it is the home host that forwards the IP packets to it. Although this home registration and the forwarding method is easy to implement and has less location registration overhead, it is inappropriate in mobile agent systems. Since all the correspondents of an agent must find its address from its home host, the agent home host may be a performance bottleneck when a larger number of agents, each with many correspondents, are originated from that same host. Besides, the agent home host may sometimes break off from the network after the agent is dispatched.

Message passing depends on the message routing and addressing mechanism. According to the addressing procedure, usually there are two message passing schemes: forwarding and locate-and-transfer. In forwarding scheme, message routing and locating a target agent are both done in a single phase. They are combined into one operation where an agent after moving to a new host informs the previous resident host where it moves so that messages can be forwarded along the extended path. The disadvantage is that the messages may take multi-hops before they reach the target agents. The performance is worsened when messages are large in size. On the other hand, locate-and-transfer locates the target agent first and then transfers the message directly to it. However, the message sender may get outdated address in cases that the receiver agent migrates frequently.

The communication mechanism in Mogent system [6] has implemented location transparency and reliable message delivery. But it also has the shortcomings of large location-updating overhead, constraints of agents' mobility and vulnerability to the address spoofing attack. All the agents need to register their locations with their homes. Before sending a message to another agent, the sender agent must query the recipient's current address from the target agent's home host. If the target agent is currently moving across the network, the reply to the location query is pending until the target agent registers its new location. Before an agent can move, it needs to ask for permission from the home host. If there are messages on their way targeting at the agent, the agent need to wait until these messages arrive. It is the responsibility of the agent home to synchronize the migration of the agents and the message passing. In this way, reliable message delivery can be guaranteed and no message forwarding is needed. However, the algorithm depends so much on the agent home that the agent cannot move and communicate if its home is down or disconnected.

In [4], mailbox-based algorithm adopts a hybrid approach combining the registration and forwarding schemes. It realizes location-transparency and ensures the message delivery. Under this communication scheme, most of the messages are sent to their receivers directly and others are forwarded at most once before they reach the target agents. Besides, the movement of agents can be separated from that of their mailboxes, thus, by deciding adaptively when to move the mailbox to its owner agent, the traffic overhead can be reduced greatly. But it is not very perfect and there are still a lot of improvements we can do to this algorithm.

In this paper the following improvements are made:

- We overcome the problem of message loss due to migration of mailbox while a message is on its way is overcome;
- We add the message-priority to the messages, so that the urgent messages get first response;
- We solve the problem of agent overloading by generating its twin;
- We update the agent automatically in a visible and predictive manner;
- We reduce the occupation on the network resources by mobile agent reconstruction and destruction, thus cleaning the agents who are idle.

III. THE ENHANCED MAIL-BOX ROUTING ALGORITHM

A. *System model and assumptions*

In our system model, we assume that mobile agent communication is largely based on asynchronous messages. This is because, when mobile agents roam the Internet, it is undesirable that two agents use synchronous communication [12], due to the large and unpredicted delays on the Internet. In [4], a mailbox is a message buffer used to store incoming messages. Every mobile agent in the system is allocated a mailbox. Incoming messages sent to the agent are inserted into the mailbox first. Two modes of message delivery can be supported: Push and Pull. In the push mode, messages stored in the mailbox will be delivered to the mobile agent, while with the pull mode, the agent fetches messages from its mailbox any time when it decides to do so. In this paper, we use the pull mode. A mobile agent is automatically initialized to check its mailbox whenever necessary. If the mailbox contains any messages, these messages are delivered. Otherwise, either a synchronous or an asynchronous receiving operation can be implemented - the mobile agent can continue its execution or is suspended until a new message arrives. We assume that the sending operation is always asynchronous (a synchronous sending can always be simulated by the means of changing the sending agent to a receiver and then making it wait for an acknowledgement after it has put the message into the message system).

In order to simplify discussion, we make some assumptions below:

- Communication by means of asynchronous message delivery.
- Fault free communication link and network host.
- No message loss or damage during its transfer.

In the scheme [4], each mobile agent has a mailbox that buffers the messages which are sent to it. As a logical part of the agent, the mailbox can be detached from its owner's agent. In other words, the agent can be located on a host different from its mailbox. When agent migrates to a new host, it can leave alone its mailbox. Figure 1 shows the communication between two agents. The rectangle represents a host. Agent A is located in Host1, Agent B is located in Host2, and Bm is the mailbox of Agent B. When A sends a message to B, it sends the message to B's mailbox Bm, and then B uses a pull operation to fetch the message from its mailbox Bm.

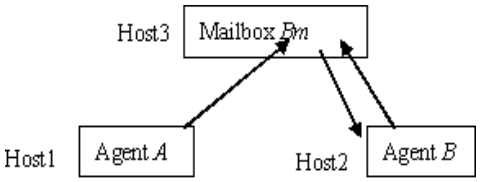

Figure 1. Mailbox-based messages forwarding scheme

B. *The mailbox relocation algorithm*

In [4], before moving, the agent determines whether it will migrate its mailbox to a new host or not. It sends a message to its mailbox host if it decides to do so. The message contains the address of the destination host where the mailbox is to migrate. On receiving the moving message, the mailbox migrates to the host specified by its owner agent. Obviously, the costs of fetching messages reduce, but the costs of the other agents sending messages to it would increase. So, at this point, it loses the advantages of detaching the mailbox from its owner agent. Besides, before the mailbox migrates, it sends 'waiting' message to all the hosts. During this span, all the agents can not send any messages to it. They have to wait until the aimed mailbox completes migrating. After it has collected the 'REPLY' messages from all the hosts or on expiration of the waiting time, it migrates to the destination host. During this time, the messages which have been sent to the old address and not to the new one and are now on the way, will not arrive at the destination mailbox. As a solution to the above problem, we propose another scheme: Improved Agent Migration Scheme.

In our scheme, whether the messages are sent at any time, they all reach the destination. In our Improved Agent Migration Scheme, before migrating, the agent performs a complex computation to measure the costs of migration and communication to determine to which host the mailbox should be moved. This scheme does not restrict the migration of the mailbox to the current agent location alone. If it decides to move its mailbox, the agent sends a message to its mailbox. If the mailbox's message queue is empty, the mailbox migrates to a specified host where the cost of the migration and the communication is minimum. Otherwise it migrates after the owner agent fetches the message queue. In our improved mailbox-based algorithm, we import the Forwarding Pointer with an element t labeled by time to avoid pointer loop. Before the mailbox reaches the new destination host, its old host maintains a Forward Pointer which points to the new host where the mailbox will locate. When the mailbox arrives at the new destination, it broadcasts an address-updating

message to all the hosts. The agent which gets a new address can send messages to the new address. Besides, if an agent has not got the new address it could send a message using the old address and the message will be sent to the former host. Because of existence of the forwarding pointer, the former host forwards the message to its destination. So even though the agent has not received the new address, it can send a message to its destination. We have seen that broadcast for location updating is inefficient. But our algorithm remains efficient. This is because as we all know, in our algorithm, agent moves frequently, while its mailbox migrates with low frequency. So the broadcast for location updating in our scheme in comparison to the other is not inefficient. On the contrary, it has its own advantage. If the receiver's location does not exist in the sender's address table or the location has been outdated, it sends a request of new address to the receiver's home. After returning the response, it updates the corresponding item of address table with the returned address. Then it sends the message to the new address. In this way, whenever the messages are sent, they all reach the destinations. This scheme effectively avoids message loss, and reduces the communication time and costs.

The mailbox relocation algorithm is depicted more formally in pseudo code:

```
MailBoxRelocation(){
string = newLocation
agent.CostsEvaluation(
every host including current
host where the mailbox locates );
//to obtain a specified host where the
//costs are minimum
if (newLocation != the host where the
mailbox current locates){
Mailbox.Currenthost.forwardPointer=
newLocation;
Mailbox.RelocateTo(newLocation);
SendUpdateAddress(newLocation, every
host including its home )
}}
```

1) Message routing algorithm

Suppose agent *A* wants to send a message to agent B, *A* first checks whether the address of B's mailbox has been cached locally. If so, it sends message to the cached address. Otherwise it sends a request message of B's address to B's home. When it receives the request, *B*'s home returns the current address of *B*'s mailbox to A. So A updates its local address table, and sends the message to the newly obtained *B*'s mailbox address. The message routing algorithm is presented more formally in pseudo code:

```
MessageRouting(MSG m) {
if (the receiver's address is in the
local address table and not outdated)
sendMsg (address in the
cache, m);
else {
String agenthomeAddress =
m.getReceiver().getHome();
sendReqToHome(agenthomeAddress, m);
String
currentmAddress=home.doRequest();
updateAddrTable(currentmAddress);
sendMsg(currentmAddress,m);
}
}
```

When a host receives a message destined to an agent M, it checks whether M's mailbox currently resides locally. If so, it inserts the message to M's mailbox directly. Otherwise, through the forwarding pointer, the message is forwarded to the next host where the mailbox is located. As mailbox migrating frequency is less than its owner agent, it is impossible to cause message chasing. So this scheme can also effectively avoid message chasing.

C. Message-sorting based on message-priority

The papers [3, 4, 8] all adopted the Mailbox-Based scheme. When arriving at mailbox, all the messages are inserted into the message queue in FIFO mode. Messages destined to an agent are all sent to the agent's mailbox, and the agent later receives the messages by either a push operation or *pull* operation. Whether it uses a push or pull operation, the agent receives the messages in a message queue. Message urgency is not taken into account at all. Agent processes the arrived messages in FIFO mode. If there is an urgent message in the tail of the message queue, it has to be processed as soon as possible. But it has to wait until the processing of all the messages ahead of it.

In this scheme, by message priority level, messages are classified into five levels i.e. 1, 2, 3, 4 and 5. The higher the level number the higher is the urgency of the message.

According to message-priority, the agent processes more urgent messages first. So this requires a message sorting mechanism. In mobile agent systems, there are two message-sorting mode(if the messages have the same priority level, message-sorting is done in FIFO mode):

1) Message sorting in mailbox: There is a message list which is dynamically maintained in the mailbox. Once a message arrives, all the messages in the list are sorted according to their priority levels. The agent fetches the sorted message from its mailbox in pull mode, and then processes them in order.

2) Message sorting in the agent: In mailbox, only a message queue is maintained. The arriving messages are stored in this message queue in FIFO mode. Owner agent fetches this message queue in pull mode. After fetching the message queue into its location, the agent sorts them by their priority levels, and then processes them in order.

In this paper, the second message-sorting mode is adopted.

D. The implementation of overloading-balance by twin agent

If there are a lot of messages that need to be processed or the receiver takes a long time to deal with one message, it is unable to fetch messages from the queue. All this leads to receiver overloading. Thus more and more messages inserted into the message queue are not fetched, and so these message senders have to wait for their response. In the case of collaboration agents they are unable to get the coordinated messages which may lead to a collaboration failure, and even cause the entire system to collapse. Accordingly, *A Mobile Agent Overloading-balance scheme* is proposed in this paper.

In this scheme, we set a timer for every mailbox to record how long the message waits in the message queue after it arrives. If any message's waiting time is beyond the specified time *Ti*, (given time), then the mailbox sends request to its associated agent's home. After receiving the request, the home constructs the target agent's twin. The little brother shares the same mailbox with its older brother. Furthermore, the twin has one more operation *Destructor()* than its older brother. The main function of the operation *Destructor()* is to destruct itself at the right time.

The algorithm works as follows: firstly, the twin checks whether a message with the waiting time beyond *Ti* exists in the mailbox, that is, whether the owner agent has fetched the messages when the twin was being constructed. If the message with waiting time beyond *Ti* doesn't exist in the mailbox and the message queue is not empty, the twin fetches the message queue to its location host, sorts and then processes the messages. After processing, the twin calls its operation *Destructor()* to destruct itself and release the occupied resources. Otherwise, the twin calls its operation *Destructor()*. In addition, if the message with waiting time beyond *Ti* exists in the mailbox, the twin fetches the message queue to its location host, sorts and then processes. After processing, the twin checks again, if the message with waiting time beyond *Ti* still exists in the mailbox. If so, it fetches, sorts and processes.

The algorithm is depicted formally in pseudo code:

```
OnAgentOverload( ) {
int count=1;
while(count >0) {
m=Twin.GetMessages(Mailbox.MsgQueue);
//Twin fetches all messages of the
//message queue from the mailbox
// of its brother
SortedList=OrderByPriority(m);
//order fetched messages by priority
if(SortedList.IsEmpty()){
Twin.Destructor();
count = 0; //return and exit
}
if(SortedList.Maxwaittime>=Ti)
Twin. ProcessMSG(SortedList);
//Twin processes the messages one
//by one in Priority order.
If(!SortedList.IsEmpty( )) {
Twin.ProcessMSG(SortedList);
Twin.Destructor();
count =0; //return and exit
}
}
```

E. The Implementation of Automatic Agent Refinement

The agents can be viewed as autonomous, problem-solving computational entities capable of effective operation in dynamic and open environments [5]. As we all know agent has certain intelligence. With the development of technology and environmental change, the capacity of the agent should become more perfect. The agent improves itself according to outside requirement and environmental changes. In order to realize the agent intelligence, we need to spent tremendous money and time. Also, the process of the agent improving itself is invisible and unpredictable to its designers and its users. This could cause unpredictable and unimaginable consequences. Here, we propose a method of gradual refinement to realize agent improvement. By this method, agent actions and behaviors are all visible and predictable.

When designing the mobile agent system, we add one more operation *Destructor()* and one max life-cycle *T*. In the course of its moving and problem-solving, agent accumulates new requirements according to environmental changes and it's perception to outside. It then sends the new requirements to its home after formatting them. Also, designers and users may add new requirements to its home. The home has a list to record the arrived new requirements. When the amount of requirements reaches a certain magnitude, the home RECONSTRUCTS a new agent with more perfect capabilities to replace the old one. Then the home informs the old agent and the old agent calls the operation *Destructor()* to destruct itself.

F. The Implementation of agent self destruction and recreation

In addition, if an agent has not received any message for a long time, and it only wanders through the network and the time it has been idle exceeds the specified value *T*, it sends a message to all the hosts including the home host and the host where its mailbox is located. When the other agents receive the message, they update the item associated with the agent in their address tables with the agent's home address. When receiving the ACK or the time out, it calls its operation *Destructor()* to destruct itself and its mailbox is also destructed by its host. If later an agent needs to collaborate with the destructed agent, it doesn't find the right address of the destructed agent, but it knows the associated home. So it sends a request to the associated home. The home RECONSTRUCTS a corresponding agent and its mailbox, and sends a new address to the requester and the other agents to update their corresponding address tables. The advantages of the scheme are: 1.able to update the old agent; 2.able to

clean the agents who are idle thereby reducing the occupation of the network resources.

IV. CONCLUSIONS AND FUTURE WORK

An improved Mailbox-Based Communications Scheme for Mobile Agent is proposed in this paper. It overcomes message loss and ensures the message delivery reliability, provides an effective solution for overloading problem of the mobile agent, ensures urgent messages getting processed first. This scheme can decrease the message waiting time effectively and also can release and reduce the resources over the network dynamically. Furthermore, the mobile agent can be updated automatically in this algorithm. By this scheme, agent actions and behaviors are all visible and predictable. Although we improved the mailbox-based mobile agent communication algorithm, there is still a lot of work needed to be done in the future. Our next step is to deal with the situation: where if a message arrives at the new location along the forwarding pointer, but the mailbox has not arrived yet? And in what circumstance the agent will prefer to move its mailbox?

ACKNOWLEDGMENT

I am endlessly grateful to our Head of the Department, respectable and beloved **Prof. N. Sankarram,** for his substantial guidance and support. I express my heartfelt thanks to my guide **Mr. P. Boominathan,** Lecturer for his inspiring dedication, untiring efforts and tremendous enthusiasm given throughout the research. His motivation, encouragement, criticism, provoking suggestions and timely help enabled us to bring out this project work successfully. My endless gratitude goes to the organizers of the conference for giving me an opportunity to present the paper. My profound and sincere thanks go to my family and classmates for their continued support in this endeavor. Last but not the least; I would like to thank the Almighty for favoring me in all situations.

REFERENCES

[1] Shengbo Chen, HuaiKou Miao and Qingguo Xu, Mailbox-Based Communications Scheme for Mobile Agent Overloading-balance and Message-priority, The Sixth International Conference on Grid and Cooperative Computing(GCC 2007)

[2] Danny B. Lange Mobile Objects and Mobile Agents: The Future of Distributed Computing? E. Jul (Ed.): ECOOP'98, LNCS 1445, pp.1 -12, 1998. ?Springer-Verlag Berlin Heidelberg 1998

[3] J. Cao, X. Feng, J. Lu, and S. Das. Mailbox-based scheme for mobile agent communications. *Computer (Computer)*

[4] J. L. J. Feng, X.Y.and Cao and H. Chan. An efficient mailbox-based algorithm for message delivery in mobile agent systems. In *Proceedings of the Fifth IEEE International Conference on Mobile Agents (MA 2001)*, volume 2240, pages 135–151, Atlanta, Georgia, DeceBer 2001. IEEE CS Press, Springer-Verlag.

[5] M. Luck, R. Ashri, and M. d'Inverno. *Agent-Based Software Development*. Artech House,INC, Boston,London, 2004.

[6] T. X.P., F. X.Y., L. X., and Z. G.Q. Communications mechanism in mogent system. *Journal of Software*, 11(8):1060– 1065, 2000.

[7] K. Verelst. *A Study of Communication Models for Mobile Multi-agent Systems*. PhD thesis, Vrije University of Brussel, Brussels, Belgium, May 1999. in chinese with English abstract.

[8] harles. E. Perkins. Ip mobility support. RFC 2002, October 1996.

[9] Z. Jia, Z. Li, and L. Xie. ACP-A Local Post Area Based Mobile Agent Communication Algorithm. *Journal of Computer Research and Development*, 41(1):47–52, Jan. 2004.

[10] M. Lunge, D.B.and Oshima. Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley, 1998.

[11] A. S. Tanenbaum and M. van Steen. *Distributed Systems Principles and Paradigms*. Prentice Hall Inc, 2002.

[12] T. X.P. *Chinese Research on Internet based mobile Agent technology and application*. PhD thesis, Nanjing University, 2001. in chinese with English abstract.