Journal of Engineering Science and Technology
Vol. 7, No. 5 (2012) 614 - 622
© School of Engineering, Taylor's University

# DEVELOPING A NEW MECHANISM FOR LOCATING AND MANAGING MOBILE AGENTS

AHMED Y. YOUSUF*, ASMA'A Y. HAMMO

College of Computer Science and Mathematics, University of Mosul, Iraq
*Corresponding Author: ayy_77@yahoo.com

**Abstract**

There is a trade-off between the agent tracking process and the message delivery process in locating mobile agent systems. In the proposed system we try to strike a balance between these two processes. Communication in Multiagent system requires an efficient mechanism to manage these communications. Reliability and transparency are the design goal for any modern mechanism. In this paper we develop a new mechanism to manage the communication among autonomous mobile agents. This mechanism has the ability to manage agent mobility in an efficient, scalable and transparent way. So no message is lost and all transitions are transparent to the user. The architecture of the proposed mechanism is described and the naming, communication, localization and re-localization ways are given.

Keywords: Mobil agent, Transparency, Localization.

## 1. Introduction

An agent is an autonomous computer system capable of reacting to and initiating changes in its environment. The feature that makes an agent more than just a process is its capability to act on its own, and in particular to take initiative were appropriate [1]. In the past few years the use of mobile agents as a model to structure distributed systems and applications have drawn a great deal of attention [2]. Mobile agents can be used to reduce network traffic by replacing communication between two components residing in different hosts in the network [3]. Multi-agent systems are being used in an increasingly wide variety of applications, like process control, which is employed in the domain of industry, system diagnostics, manufacturing, transportation logistics and network management [4]. One of the most important challenges in mobile applications is communication. This problem has been discussed extensively in the area of distributed intelligence and multi-agent systems [5]. Any designed approach for location transparent communication must provide solutions for [4]:

- Agent tracking: keep track of current agent location to make it possible to find the agent.
- Message delivery: the message sent to the mobile agent must not be lost; it must be delivered to the current agent location.

Two approaches have been proposed [4]:

- Full information approach: In this method every agency has full knowledge about the current location of all agents in the system. For each migration, all agencies in the network must update their local location directory. So agent tracking becomes a very expensive task, while, message delivery is an easy task, like the broadcast method.
- No information approach: No agent tracking is established in this method, therefore no agency has direct knowledge of the current location of any agent other than those residing locally. Agent migration is very cheap but message delivery is very expensive, like Forwarding pointer.

In this paper we combine between these methods and focus on the problem of providing location-transparent communication, that is, on how two or more mobile agents that are frequently migrating can communicate with each other in a reliable way. By transparency we mean that the applications should not include localization-related instructions. The localization and re-localization processes are implemented within the underlying mobile agent system. The programmers of mobile agents-based applications should not include code to locate agents explicitly. Reliability considers message tracking. A message can be sent to an agent that moved to another location, or is in the process of moving. The message tracking service consists of the mechanisms which allow delivering messages to their destination. Messages should not be lost.

This paper is organized as follow; Section 2 discusses the related works. Section 3 explains the proposed Mobil agent locating mechanism and its details. Finally the conclusions are shown in Section 4.

## 2. Related Work

The research in the area of mobile agents programming is highly active. In Telescript, the proposed local communication is provided by Call specific method called meet which define the target agent, while global communication is implemented through connections, which are point to point connections between agents residing in different agencies [6]. Agent Tcl uses the Remote Procedure Call (RPC) method to allow remote communication between agents. The programmer had to implement his own techniques for tracking mobile agents [7]. Aganta, a Java based mobile agent tool kit, provides Remote Method Invocation (RMI) [8].

In Mole, messages received by a mobile agent site and which are intended for a mobile agent in process of moving are not accepted. Mole does not suggest solutions for the communication links between the mobile agent and its clients

[9]. Management of communication links become the responsibility of the applications agents.

MOA provides another approach for the mobility management of agents. The approach is based on the migration of the communication channels established between the client agents and the server agent which moves and with which the clients communicate. Client agents receive an exception message for every communication with an agent whose movement has been initiated. It is the programmer’s responsibility to deal with these exceptions messages, i.e., by locating the mobile agent and re-sending the messages [10].

## 3. The Proposed Locating Mobile Agent Mechanism

There is a trade-off between the agent tracking process and the message delivery process in locating mobile agent systems. In the proposed system we try to strike a balance between these two processes. We use the term "client agent" to refer to the agent that requests the services and the term "server agent" to refer to the agent that provides services.

The proposed locating mechanism has the following features:

- For transparency in maintaining communication links via a location-transparent communication service, the client agent must be able to communicate with the mobile agent despite of its mobility.
- Reliability: Guarantee the delivery of messages to the mobile agent even as it moves.
- Scalability: The mechanism must scale well, even with large networks, and it must not reduce efficiency.

### 3.1. The proposed system architecture

In the proposed system we view the internet as consisting of regions. Each region contains a central server which in turn contains two directories, the Global Agent Location Table (GALT) and the Local Agent Location Table (LALT).

The former is used to keep track of agent location in other regions, the latter is used to keep track of agent location inside current region. GALT consists of three fields. The first field is the agent name, the second field is the destination region name, the third field is the status field which can contain one of two values, 0 or 1, the 1 value indicates that the agent is currently in the destination region, and the 0 value indicate that the agent is in movable state. LALT consists of two fields: agent name and destination site name.

Each region consists of a number of sites; these sites are considered an execution environment for creating, executing, suspending and destroying an agent. The messaging system which is a part of each execution environment is responsible for establishing a communication link between agents both locally and remotely. It is also aware of the status of the links; if they are broken, then it takes the necessary action to establish new links. Finally, the agent which is a mobile object can move from site to site. Figure 1 shows the overall architecture of the proposed system and Fig. 2 shows the site components.

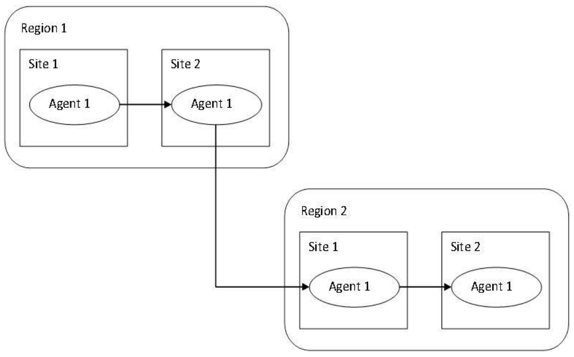

**Fig. 1. General System Architecture.**

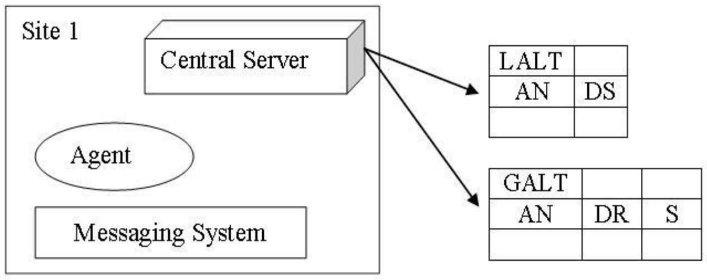

**Fig. 2. Site Components.**

### 3.2. Naming

We use a global naming schema. The agent name has the following format:
Region number: $hash_{64}$ (site address || id). The region number is used to designate the central server of the generated agent. $Hash_{64}$ (site address || id) is used to generate a 64-bit unique code to identify the agent name. When the agent is first created, it is registered with the unique name at the name server, and then it is registered inside the LALT at its region. If the agent moves to another region, the new location is registered in the GALT and the status is set to 0 until the agent reaches the destination. Figure 3 describes the agent registration process.

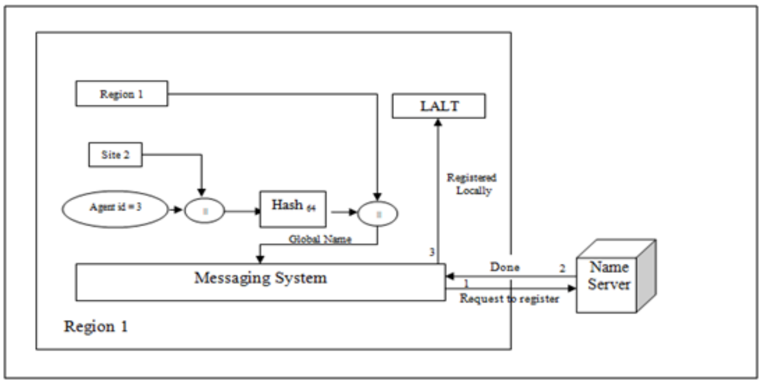

**Fig. 3. Naming Schema.**

## 3.3. Agent communication

The client agent communicates remotely with the server agent using Remote Method Invocation (RMI). To establish a link between client agent and server agent for the first time, the client agent gets a server agent global name transparently from the name server.

The global name contains the region name for the server agent, which is also the name of the central server in that region. If the client agent wishes to re-establish a broken link with the server agent, then the client agent gets the current location from GALT in its region.

## 3.4. Mobile agent localization

When a client agent wishes to communicate with a mobile server agent, the messages go through the client messaging system. The client messaging system will then send a localization request to see if the central server at the same region. The central server will check if the server agent is co-located in its region using LALT to establish a local communication. If it is not, then it will check it in the GALT and establish remote communication. The algorithm below outlines the necessary steps for the mobile agent localization process:

```
Step 1: Client agent's messaging system generates a communication request.
Step 2: Send the request to the central server in the region.
Step 3: Check to see if the request is in LALT then
                Establish communication
        Else
                Check to see if the request is in GALT and establish
communication
```

## 3.5. Agent re-localization

As we mentioned before, the agent global name contains the name of the region where the agent was first created. This name is also the name of the central server for this region. If a client messaging system wishes to communicate with the server agent messaging system which has been moved, then the re-localization process is as follow:

The client agent messaging system gets the location of the server agent which is out of date from its central server in its region. When the client agent messaging system sends a message to the server agent messaging system, then an exception message is returned to the client messaging system and a new location message is sent to the central server of the client agent to update its GALT. The central server in turn will inform the client agent messaging system about the new location to establish a connection. Figure 4 shows this process. The sequence of steps in Fig. 4 is as follows:

- The messaging system of the client agent uses the old address of the server agent to establish communication.
- The messaging system of the server agent sends an exception to the client agent.
- The central server of the old region sends the new location of the server agent to the central server of client agent.

- The central server of the client agent updates its GALT and informs the client agent messaging system about the new location of the client server.
- The client agent messaging system uses the new location to establish communication with the client severs.

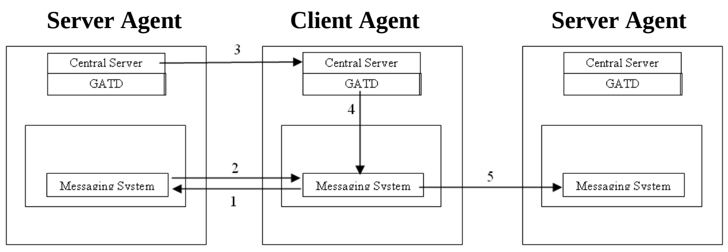

**Fig. 4. Re-localization Process.**

## 3.6. Agent mobility

When the server agent moves to another region, then all the communication links will be broken. The sent message must not be lost, so when the server agent moves, it informs the central server of the current region about the next destination region. If the client agent messaging system sends a message to the messaging system of the server agent which is in a movable state, then an exception message is returned to the client agent messaging system to suspend sending messages. The same scenario will be repeated if a new client agent wants to communicate with the server agent. When the server agent reaches the destination, it sends back an acknowledgement to the central server at the region where it was before. Upon receiving the acknowledgment, the central server at this region will modify the GADT with the new location, and then the acknowledgment will be sent to all broken links or new request links by one of three methods:

**First method:** The central servers of the receiving region will advertise the new location in a multicast message to all interested regions. The central server of the receiving region will do the following algorithm according to method 1:

***Step 1**: Generate a message with the new location of the server agent.*
***Step 2**: Multicast the message to all regions listed in the region list.*

The central server of the interested region will do the following:

***Step 1**: Modify the GALT.*
***Step 2**: Establish communication.*
***Step 3**: Inform all client agents' messaging systems with broken links or requests for new links about the new location.*

**Second Method:** Advertise to all interested regions by a method called "forward to nearest region". The advertising message contains the new location and a list of all interested regions. Each region receiving the message will remove its name from the list and forward it to the nearest interested region. The central server of the receiving region will do the following algorithm according to method 2:

```
Step 1: Generate a message that contains the new location with a list of all interested regions.
Step 2: Send the message to the nearest interested regions.
```

The receiving region will do the following:

```
For each receiving region:
  1: Update GALT.
  2: Remove the region name from the received message.
  3: Send the modified message to the nearest interested region.
```

**Third method:** Advertise all regions by a method called "forward to all" using successor and predecessor order. In this method, each region has a number so the regions are numbered from (1-*n*). If the server agent advertises the new location to the previous region with number 5, for example, then the region will forward two messages about the new location. The first message will be sent to the successor of the region, which is region number 4, then region 4 will forward the message to region 3, and so on until the messages reaches region 1.

Region 5 will also forward the message to region 6, region 6 in turn will forward the message to region 7, and so on until the message reaches region n. The received region knows to which region it should send the message by checking the number of the forwarding region. If the number is less than its number, then it must forward it to the predecessor region; otherwise, it must forward it to the successor region.

The following algorithms describe method 3:

```
Step 1: Set $x$ = region number
Step 2: If ($x > 1$ and $x < n$) then
        1: Generate two messages with new location.
        2: Send first message to region with number ($x – 1$).
        3: Send second message to region with number ($x + 1$).
            Else
Step 3: If ($x = 1$) then
        1: Generate message with new location.
        2: Send message to region with number ($x + 1$).
```

The receiving region will do the following:

```
Step 1: Update GALT in the central server.
Step 2: $x$ = region number
Step 3: If send region number > $x$ then
        Send the message to region ($x – 1$).
            Else
        Send the message to region ($x + 1$).
```

In all methods the receiving region will modify the GALT with the new location in its central server, and then the central server will check to see if there is a client agent in its region that needs to communicate with the server agent to inform it of the location and establish a connection.

The following algorithm describes how the region of a movable server agent treats an incoming message:

```
Step 1: If the message is acknowledged by the mobile agent then
            1: Update the GALT with the  new location.
            2: Send the new location message to the  interested region using method
               1, method 2    or method 3.
                        Else
Step 2: If the message is from a  broken communication link then
            Send an exception message to the broken communication link.
                        Else
Step 3: If  the message is a new communication request then
            1: Send an exception message to that communication link.
            2: Add the region name to the list of regions that must be informed about
               the new location of the agent.
```

## 3.7. Message reception

When a server agent messaging system receives a message from a client agent messaging system, then the central server of server agent will check the state of the server agent. Three cases can be found:

- The server agent is in a movable state, in which case the central server will implement the agent in mobility state process.
- The server agent is in a stable state, with the agent inside the region of server agent, in which case the message is forwarded to the server agent messaging system.
- The server agent is moving, in which case the central server of the region will implement the localization process.

The following algorithm outlines the message reception process

```
Step 1: If  the server agent is in a movable state then
            Implement the agent in mobility state process.
                        Else
Step 2: If the server agent is in a stable state then
            1: Get its current location from LALT.
            2: Send a message to the server agent messaging system
                        Else
Step 3: Implement re-localization process
```

## 4. Conclusions

In this paper, a mechanism for the localization of mobile agents and the management of messages is proposed. A global naming schema for the agent name is used and we propose three different methods for the management of agent mobility, which considered efficient and transparent to client application. We provide transparency in maintaining communication links via transparent localization. The mechanism for the management of messages is devised in a way to minimize the management overhead and provides efficiency.

## References

1. Wooldridge, M. (2009). *An introduction to multiagent systems.* (1st Ed.), John Wiley & Sons Ltd.
2. Harrison, C.; Chess, D.; and Kershenbaum A. (1995). Mobile agents: Are they a good idea? *Research Report*, IBM T.J. Waston Research Center.
3. Murphy, A.L.; and Picco, G.P. (2002). Reliable communication for highly mobile agents. *Autonomous Agents and Multi-Agent Systems*, 5(1), 81-100.
4. Bellifemine, F.L.; Caire, G.; and Greenwood, D. (2007). *Developing multi-agent systems with JADE.* (1st Ed.), John Wiley and Sons Ltd.
5. Glaser, N. (2002). *Conceptual modelling of multi-agent systems: The ComoMAS engineering environment*. Kluwer Academic Publishers.
6. Braun, P.; and Rossak, W. (2004). *Mobile agents: Basic concepts, mobility models, and the Tracy toolkit*. Morgan Kaufmann.
7. Gray, R.S. (1996). Agent Tcl: A flexible and secure mobile-agent system. *Proceedings of Fourth Annual Usenix Tcl/Tk Workshop*, 9-23.
8. Karnik, N.M.; and Tripathi, A.R. (2001). Security in the Ajanta mobile agent programming system. *Software: Practice and Experience*, 31(4), 301-329.
9. Baumann, J.; Hohl, F.; Rothermel, K.; and Strasser, M. (1998). Mole - Concepts of a mobile agents system. *World Wide Web*, 1(3), 123-137.
10. Milojici, D.S.; LaForge, W.; and Chauhan, D. (1998). Mobile objects and agents (MOA). *Distributed Systems Engineering*, 5(4), 214-227.