JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2007**

Vol. 6, No. 4, May-June 2007

Increasing Mobile Agent Performance by Using Free Areas Mechanism

Dr. Tarig Mohamed Ahmed, Department of Computer Science, King Faisal University, KSA

Abstract

One of the most important issues that measures a quality of distributed system applications is performance. Mobile agent applications have a high performance by comparing with client-server applications. But, these applications may face some problems in the performance. The problems may relate to the agent size. The mobile agent size may increase during a journey. Some hosts may reject to receive the mobile agent in case it has a large size. Also, a mobility of the agent consumes more time and increases the network traffic. This paper proposes a new approach for such problem by introducing Free Area Mechanism (FAM). FAM is a new mechanism that reduces the mobile agent size during its journey. As result of reducing the agent size, the performance will be improved and the agent will be more accepted to the hosts. The main idea behind this mechanism is to free some useless parts from the agent body. By using .Net Framework and C# language, a prototype has been developed. Through the prototype, two experiments have been accomplished with and without FAM and the results have been analyzed.

Keywords: Distributed system, Performance, Mobile Agent system.

1 INTRODUCTION

A mobile agent technology is an autonomous piece of code that can migrate under self-control from one machine to another in a heterogeneous network. This technology is sub area of the distributed system and it is a new method for computers to communicate. Mobile agent systems provide a good infrastructure to implement many distributed application systems. It contributes in developing a method for efficient computing and information searching and retrieval because; the agent migrates to a host where the date is located instead of transmitting data across the network. This way reduces network traffic and overcomes network latencies. In addition, it supports fault tolerant and robustness capabilities of the applications. Traditional models in distributed systems may use client-server paradigm which clients and servers communicate either through message passing or remote procedure call (RPC) [1, 2]. This communication method is usually synchronous and it may consume a long time because, users must be online during the

Cite this article as follows: Tarig Mohamed Ahmed: "Increasing Mobile Agent Performance by Using Free Areas Mechanism", in *Journal of Object Technology*, vol. 6, no. 4, May-June 2007, pp. 125-140 http://www.jot.fm/issues/issue_2007_05/article4

communication. A mobile agent model uses a remote programming method (asynchronous). This method allows the mobile agent to execute its task in hosts (remote machines). This way of communication reduces the communication and connection time because users may be offline during agent roams among hosts. So, mobile agent systems have a high performance compared with client server applications. On the other hand, the mobile agent system may face some problems that decrease its efficiency. Baek et al [3] mentioned that there are two factors which affect the mobile agent performance: (1) the number of mobile agents and (2) the total of routing time taken by the participant agents. Also, the size of data that is transmitted across the network definitely affects the cost of transmission. The large data needs more time. No doubt, a mobile agent's size influences the cost of the agent's journey. So, a large agent consumes more time than a small one. From this point, the agent's size must be considered when we discuss the mobile agent performance. It is important to reduce the agent's size in order to increase the mobile agent performance.

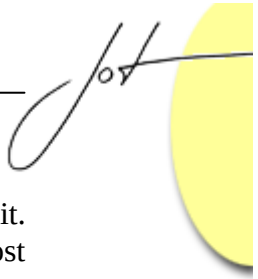
This paper proposes Free Area Mechanism (FAM). FAM is a new mechanism that reduces the agent's size during a journey. The main idea behind this mechanism is to eliminate unused parts from the agent during its journey. By this way, the agent's size is reduced and dramatically, the performance will be increased. In addition to the increase of performance, FAM lets the agent accept new behaviour that allows the agent to visit new detected places that are not scheduled in the agent's itinerary. Moreover, this mechanism contributes in solving the problem of the host that prevents to accept the visitor agent which has a large size. FAM has been implemented in this paper and some experiments have been done with and without FAM to evaluate the efficiency of the mechanism. Based on experiments result, the agent performance has been analyzed and discussed. The paper is organized as follows. Section 2 presents a background on the mobile agent's components and some relate work on the mobile agent performance. Section 3 offers a detailed presentation of FAM architecture. Section 4 presents FAM implementation and experimentation and the results analysis. As a conclusion, Section 5 presents a summary of paper and some indication for future research.

2 BACKGROUND AND RELATED WORK

This section describes the basic concepts about mobile agent system. Also, some interesting researches are presented**.**

Mobile agent: A mobile agent is an entity that works on behalf of its owner to perform different tasks at several places. It can move from one place to another under self-control. From a programmer perspective, the mobile agent is an executed code that can carry a list of computed operations to perform on different machines and collect results from these machines.

Mobile agent base: A mobile agent base is a home of mobile agents. From this component, the mobile agent starts its journey. When it completes its duty, it returns home with the results.



- Host**:** A host represents a machine (Service provider) that the mobile agents visit. It provides the mobile agents with resources. According to its policies, the host provides the agent with services. Also it can serve many mobile agents simultaneously.
- Mobility**:** This is a key feature of the mobile agent system. Through a mobility mechanism, the mobile agent can migrate from one place to another in a heterogonous network. After achieving its goals in the host, the agent requests the current host to dispatch it to the next place (host or home). According to the request of the agent, the host captures mobile agent states. By using mobility mechanism, the agent migrates to the target destination. Vigna and Fuggetta et al.[4, 5, 6] defined the agent components that are hosted by the host as the execution and resource units. The execution unit represents an algorithm of the computations. The resource represents the entities that provide the algorithm with information. Most mobile agent systems offer two forms: (1) Strong mobility that allows a code and an execution state of the execution unit to migrate to different places. (2) Weak mobility that allows a code with some initial value to migrate without an execution state.
- Communication: Communication messages allow the components of the mobile agent system to communicate. The communication model can be a synchronous communication or an asynchronous communication. Many mechanisms are introduced in this area and they are detailed in [7, 8, 9, 10], such as: Message Passing, Methods Invocation, Collective Communication, Data Shared, Event Message, Session Communication and KQML Communication.

On the other hand, many interesting researches are investigated the performance of the mobile agent by using different parameter measures. The rest of this section presents with some of these researches:

- Baek et al [3] proposed planning algorithms tried to find the minimum number of agents, and the total routing cost consumed by those agents while prevent the total execution time from exceeding the minimum. They had stated that there are two significant planning factors affecting the performance of the agent system in the network environment that are the mobile agent's itinerary and the number of the mobile agent. The experiment of this research proves that if the size of a mobile agent is begin increased while retrieval operations are performed, the bandwidth varies from link to link. In this case, the agent will consume more time.
- Cook [11] has mentioned that building software system composed of mobile agents introduces interesting new concerns for software engineering research. He described some assumptions behind mobile agent systems and software engineering. One of them: Code is cheaper to move than data and this assumption implies that the size of the mobile agent is important and it should be reduced.
- Dikaiakos et al [12] proposed a hierarchical framework for the quantitative performance evaluation of mobile-agent middleware platforms. The framework for benchmarking is developed to enable the performance characterization of key

components of the mobile agent middleware, and analyzed the performance of important classes of mobile agent applications. The parameters that have been used in the benchmark are type of operating systems (UNIX, Windows), channel configuration (LAN, WAN), the size of mobile agent and the number of times the benchmark are executed.

- Ismail et al [13] have analyzed the mobile agent paradigm. The analysis presents a performance evaluation of the mobile agent paradigm in comparison to the client /server paradigm. Two applications have been selected to the application domain that targeted by mobile agents. These applications have been implemented by three ways, first using Java-RMI (which implements client/server paradigm), second using a prototype and finally using the Aglet mobile agent system. The results of this experiment show that the use of mobile agent systems can lead to significant performance improvements.

3 FREE AREA MECHANISM (FAM)

The main goal of FAM is to increase the mobile agent performance. After the mobile agent completes a part of its journey, this mechanism can reduce a mobile agent size by freeing some unused parts from its body. When a mobile agent data is reduced, automatically the mobility of the agent consumes less time. In addition, the mobile agent size will be more accepted to the hosts. No doubt, all this factors improve the general performance of the mobile agent system.

As defined above, the mobile agent is composed of several components. These components represent tasks at each place. After the agent completes a part of its tasks during a journey, some of its components are not be used in the journey rest. Therefore, these components are overhead to the mobile agent. FAM frees these components to minimize the mobile agent size without any side effects to the mobile agent behaviour. The benefits of this mechanism are clearly appeared in case it implemented with mobile agents that perform different tasks in the hosts. In addition, this mechanism is perfect when it applies to a mobile agent system that uses the strong mobility mechanism.

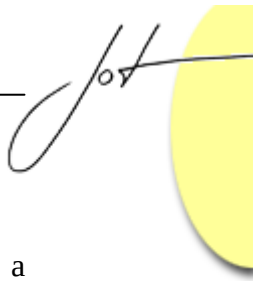
FAM may be implemented in an insecure or secure environment. As follows, some details are given to describe FAM architectures in two environments**.**

FAM architecture in an insecure environment

An insecure environment may contain some hosts that might attack the mobile agent during the reducing size process.

FAM has some assumptions that must be in our consideration in order to get its benefits:

- Mobile agents visit many places in a journey.
- Mobile agents may perform different tasks in those places.



- Mobile agent system uses strong mobility mechanism.
- Mobile agent system has many controllers (machines) that are distributed in a system domain.
- The controller is a safe place and the main role of this place is to reduce mobile agent size (Location for decreasing weight).

FAM consists of several components each one has specific role as follows:

Agent catalog

This component contains information about elements that are included in the mobile agent and their status. As an abstract view, the mobile agent consists of variables (data) and methods (codes). We can see the catalog as a table with three fields: Element ID, Place and Status. The Element ID represents a method or a variable ID. The Place specifies on which places the element will be used. The Status takes value ON in case the agent still needs this element and value OFF in case the element will not use again. Each component in the mobile agent is represented as record in the agent catalog. Agent catalog is distributed over mobile agent body. According to mobile agent content, the mobile agent base creates and encrypts Agent catalog by using Symmetric Encryption mechanism . Only controllers can deal with Agent catalog for security reasons. No host can deal with the agent catalog in order to protect the catalog against any attack (deletion or modification).

Controller

A controller is a safe place for reducing an agent's size. The main aim of this place is to perform the operation of reducing size securely . The mobile agent system distributes some controllers around hosts. The mobile agent can visit many controllers during its journey. After the mobile agent visits some hosts, it may migrate to the controller in order to free some unused parts of the agent. Each controller in the system knows a secret key that is used in the agent catalog encryption. To achieve its duties, This place introduces some services to the visitor agents as follows:

- Decrypting Agent catalog with the secret key and according to journey history records, it updates Agent catalog status.
- Specifying all elements that will not be used in the journey rest and assigning them as deleted elements.
- Rebuilding the mobile agent in a new form by eliminating the deleted elements.
- Rebuilding a new version of the agent catalog according to the new mobile agent form.
- Encrypting Agent catalog again with the secret key.
- Enclosing Agent catalog with the mobile agent.
- Allowing the mobile agent to continue its journey.

To achieve its duties, the controller uses some units as follows:

Garbage collection unit (GCU)

This unit uses a mobile agent, an agent's itinerary and an agent catalog to specify the deleted elements that should be deleted to reduce the agent's size.

Agent rebuild unit (ARU)

After GCU specifies the deleted elements, the ARU removes all these elements from the mobile agent and rebuilds a new version of it. Also, it updates the agent catalog according to existent agent elements. So, this operation reduces the mobile agent size.

Results Summary Unit (RSU)

Before the mobile agent arrives to the controller, it visits some hosts. So, it may contain some result. RSU can carry out and make a summary of these results. This way allows the controller to assign more agent elements as deleted elements.

Dynamic Behaviour Unit (DBU)

In some situations, mobile agents may be interested in visiting new places that are not scheduled in their itinerary tables. No doubt, these agents need behaviours that allow them to deal with these places. DBU can play this role as dynamic behaviour provider according to the mobile agents' requests.

Communication Unit (CU)

The role of this unit is to receive and dispatch mobile agents to/ from the controller. Also, it can provide an agent's owner with some urgent results. CU allows the controller to receive many agents simultaneously through multithread paths.

Security Unit (SU)

Most mobile agent systems encrypt their mobile agent for security reasons. Also, Agent catalog is moved in encryption form. SU can be used to decrypt the mobile agent and the agent catalog when the mobile agent arrives to the controller. It can be used also to encrypt them when the mobile agent leaves the controller.

Figure 1 shows FAM in a mobile agent system. Figure 2 shows the controller architecture.

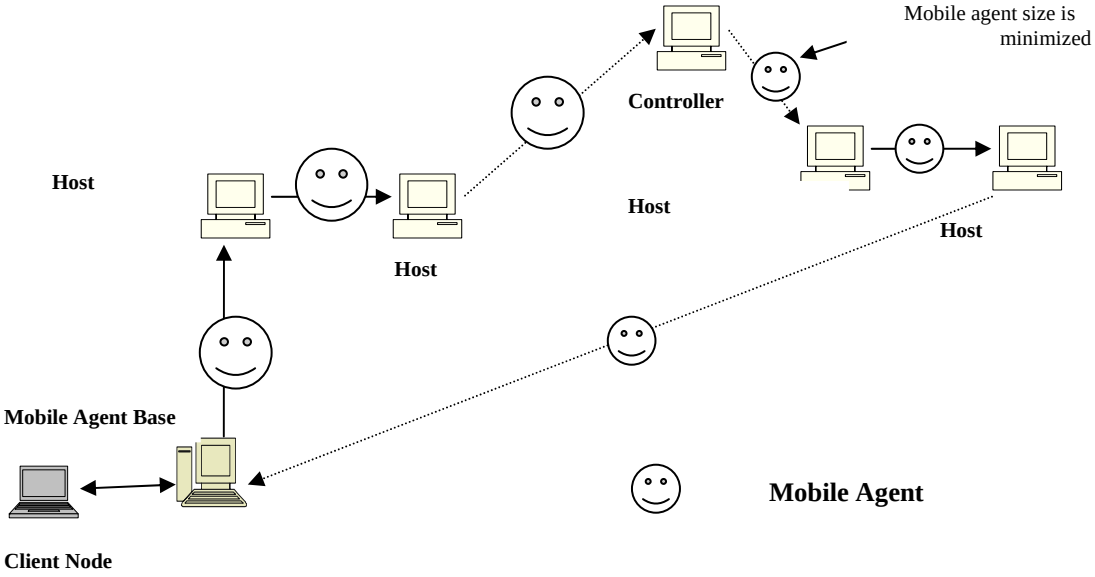


Figure 1. FAM architecture

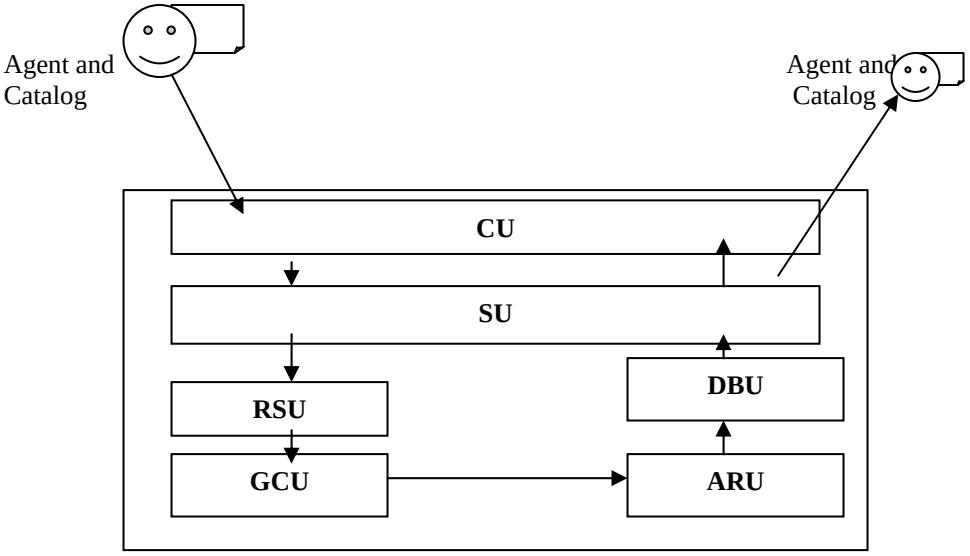


Figure 2 Controller architecture

FAM architecture in a secure environment

By setting a copy of all controller components that are described above in each host, FAM can be implemented in a secure environment. It is important to be sure that the agent catalog is secured in each host.. Before the agent leaves its current host, it will visit the controller components that are hosted by the host. These components reduce the size of the mobile agent as detailed above. This process is applied to the agent in each place.

No doubt, the agent size average is minimized in the secure environment during the journey more than the insecure environment. But, there is an overhead of the reducing process cost. In the following section, some experiments have been done to present advantages and disadvantages of FAM.

4 FAM IMPLEMENTATION AND EXPERIMENTATIONS

Implementation

To give some sort of trust to our proposed mechanism and to ensure its efficiency, a system prototype has been developed by using .Net Framework and C# language. FAM is included in the implementation. The prototype consists of a user interface, a mobile agent base, hosts and controller. The agent base creates the mobile agent and the agent catalog according to user's specification. By using the agent itinerary table, the agent starts its journey. During the journey, the agent visits some hosts and the controller. LAN, TCP/IP protocol and Windows XP are used as infrastructure of our prototype.

Experimentations in the insecure environment

Two experiments have been done to test the prototype. In the experiments, the mobile agent visits four hosts and it retrieves a book price from each one. The two experiments have used the same mobile agent but by different ways. In The first experiment, the agent visits only the hosts (without FAM). In the second experiment, the agent visits the hosts and controller (with FAM). The results that conducted by the experiments are as follows:

Experiment No. 1

According to the text in figure 3, the mobile agent base creates a mobile agent. The agent will visit Host1, Host2, Host3 and Host4. It looks for a book price in each one. After completing its journey, the agent returns to home. Table 1 presents the mobile agent size during the journey.

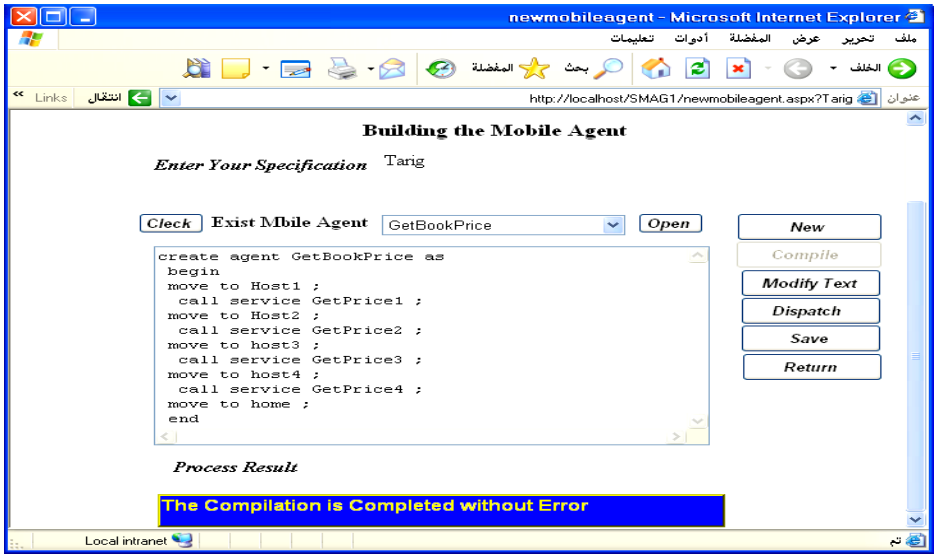


Figure 3 the Mobile Agent Description

Machine	Bytes
Home	7328
Host1	7408
Host2	7493
Host3	7569
Host4	7633
Average	7486.2

Table 1 Agent size during the Journey

Experiment No. 2

According to the text in figure 4, the mobile agent base creates a mobile agent. The agent will visit Host1, Host2, Controller, Host3 and Host4. It looks for a book price in each one. After completing its journey, the agent returns to home. Table 2 presents the mobile agent size during the journey.

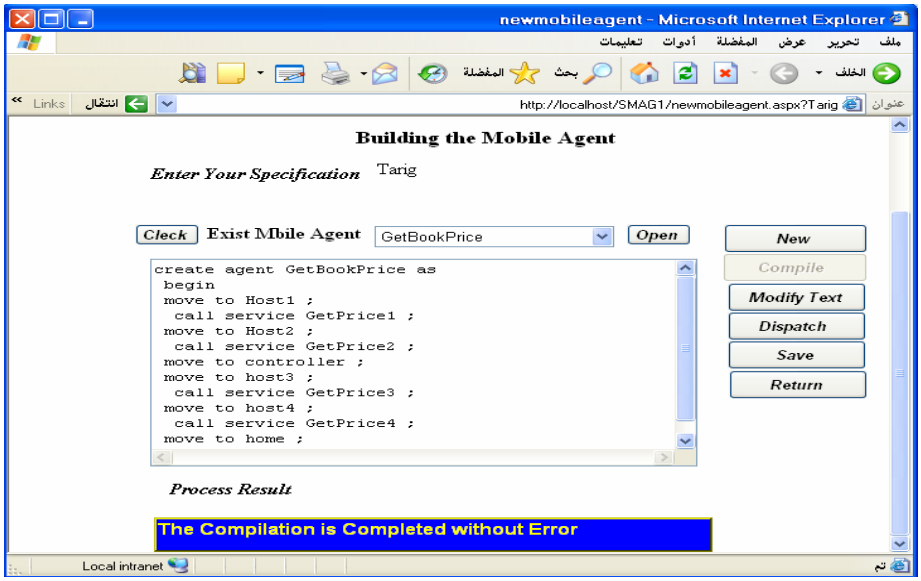


Figure 4 the Mobile Agent Description

Machine	Bytes
Home	7920
Host1	8000
Host2	8085
Controller	4643
Host3	4705
Host4	4785
Average	6356.3

Table 2 Agent size during the journey in the insecure environment

Results analysis

Experiment No. 1: as showed in Table 1, the mobile agent starts with 7328 bytes. The size is increased 305 bytes (%4.2) during the journey and that is due to the execution processes among hosts. The average size during the journey is 7486.2 bytes. Figure 5 shows the chart of this experiment.

Experiment No. 2: as showed in Table 2, the mobile agent starts with 7920 bytes. The agent visits additional place (controller). Therefore, the agent size is a larger than the agent size in the experiment No1. The size is reduced 3135 bytes (%49.3) during the journey and that is due to using FAM in the controller. The average size during the journey is 6356.3 bytes. Figure 6 shows the chart of this experiment.

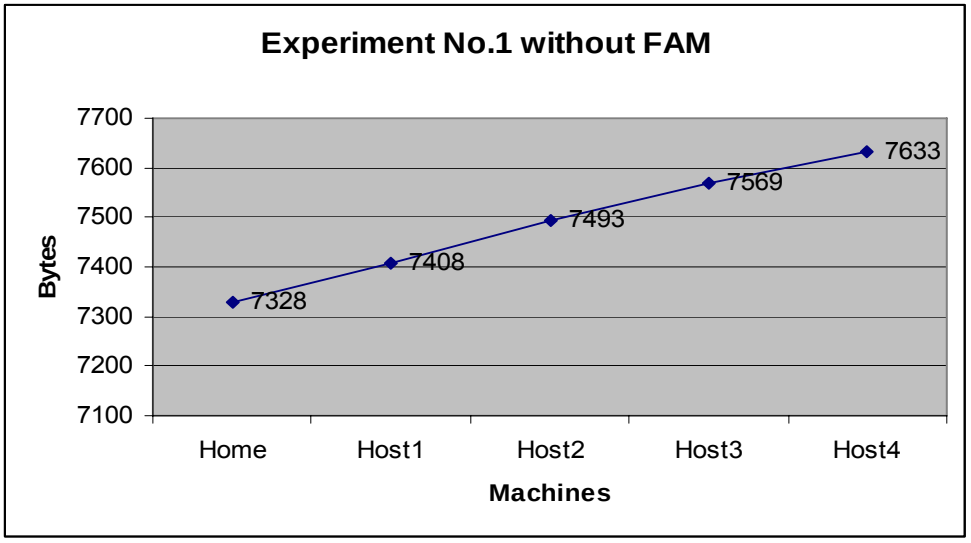


Figure 5 Experiment No.1

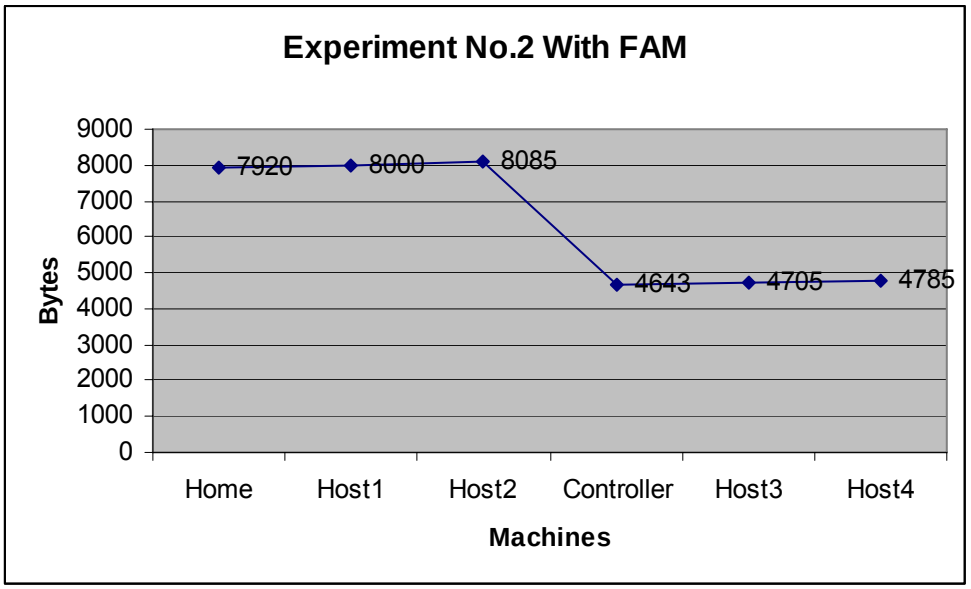


Figure 6 Experiment No.2

Figure 7 shows the two experiments (for the hosts only).

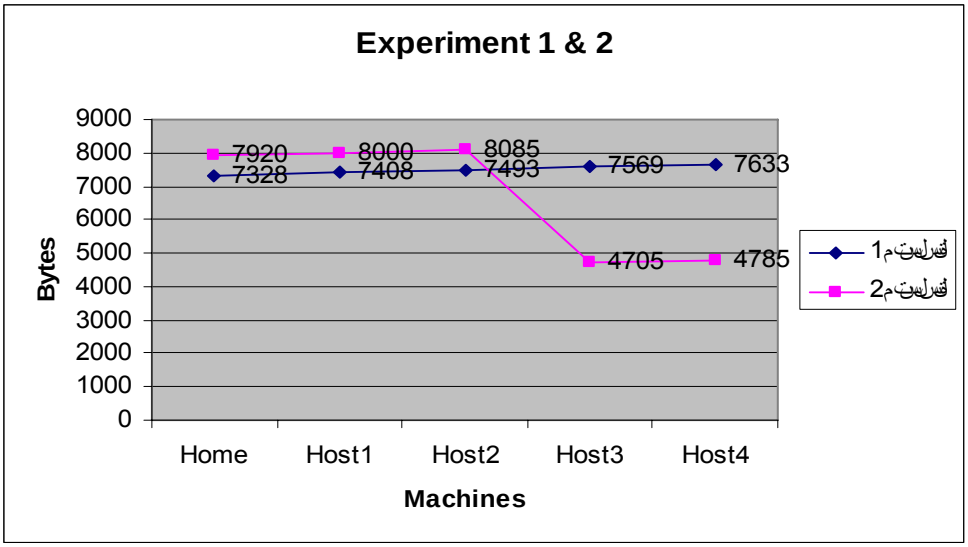


Figure 7 Experiment No.7

According to the results of the two experiments, no doubt That FAM reduces the size of the mobile agent when it works in the insecure environment.

Experimentations in the secure environment

Experiment No.3

To implement this experiment, the controllers have been removed from the system and a copy of the controller components are installed in each host in the system. The agent described in figure 3 is generated again to be executed in the secure environment. Table 3 presents the mobile agent size during the journey.

Machine	Bytes
Home	7328
Host1	5140
Host2	4005
Host3	2897
Host4	1825
Average	4239

Table 2 Agent size during the journey in the secure environment

Results analysis

As showed in Table 3, the mobile agent starts with 7328 bytes. The size is reduced 5503 bytes (%75.1) during the journey. Therefore, FAM reduces the mobile agent size in the

secure environment more than the insecure environment Figure 8 shows this experiment and figure 9 shows experiment No.1 without FAM and this experiment.

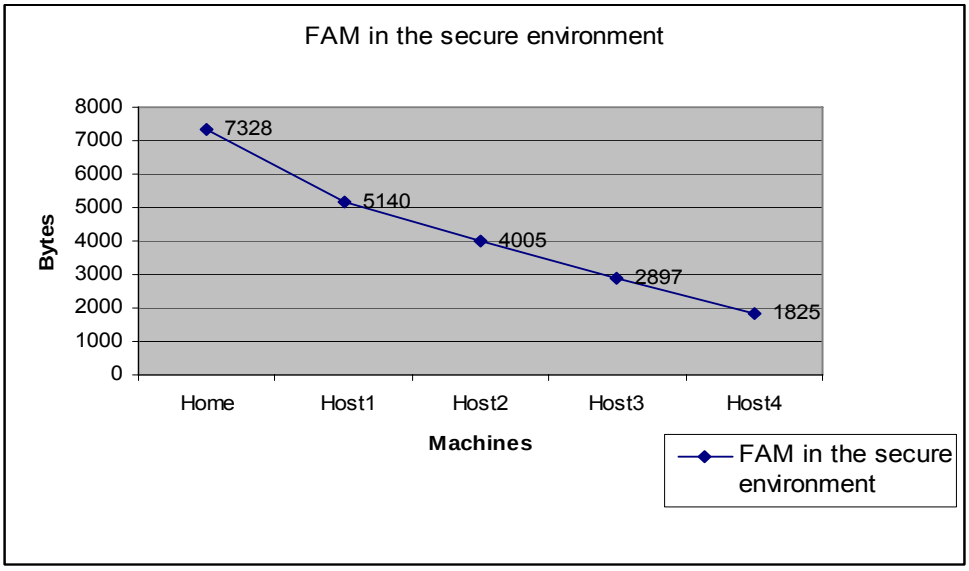


Figure 8 FAM in the secure environment

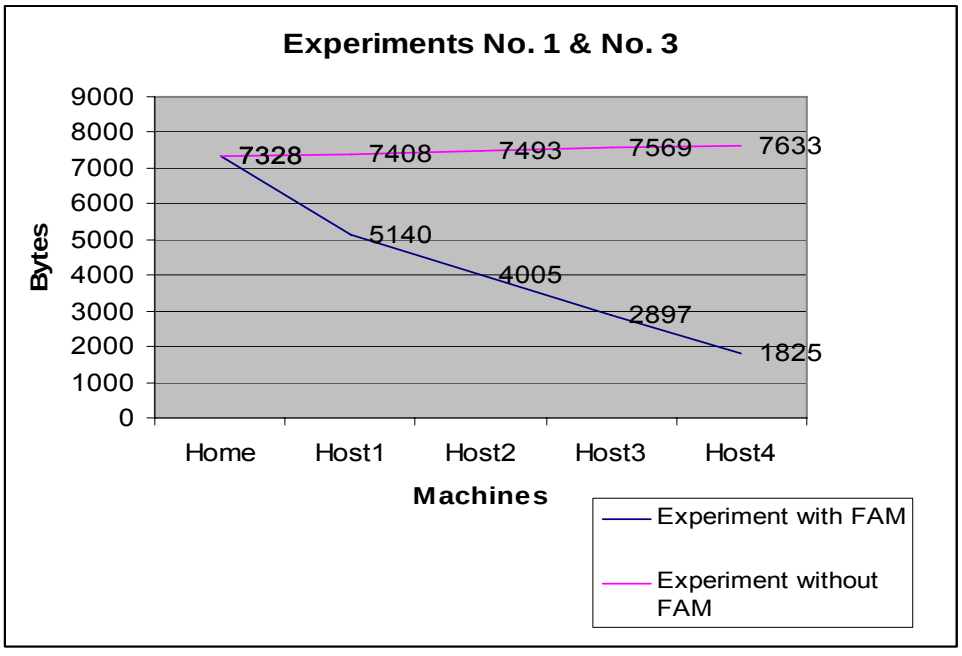


Figure 8 Experiments No.1& No. 3

Time cost

No, doubt, the total time cost of a mobile agent journey is important in the performance area. According to the three experiments, the total time cost of each one is measured and they are as follows:

a) Experiment No. 1: Total cost time of the journey equals 2015 ms
b) Experiment No. 2: Total cost time of the journey equals 2593 ms
c) Experiment No. 3: Total cost time of the journey equals 1968 ms

As shown, the total time cost in the experiment No. 2 is more than the others and that due to the agent visits the controller this takes additional time. On the other hand, the total time cost of experiment No.3 is less than the others and that because the mobile agent size is reduced during the journey.

5 CONCLUSION AND FUTURE WORKS

This paper described FAM as a new mechanism that increases the mobile agent performance securely. The main idea behind this mechanism is to reduce the mobile agent size. FAM establishes a good environment for mobile agent systems that allows the mobile agents to visit new detected places during a journey. The mobile agents can accept new components that represented behaviours in the new places. Some hosts refuse the mobile agents because their sizes are large and not acceptable to store them. FAM can help in this situation. According to FAM architecture, the mobile agent system prototyping has been implemented by using .Net framework and c# language. Two experiments have been performed and according to the result, FAM proved its efficiency in reducing the mobile agent size.

As future work and to take advantage of FAM, this mechanism can be implemented in several existent mobile agent systems that put the assumption of FAM in their consideration. So, we can measure the benefits of FAM and no doubt it will open new approaches to enhance mobile agent models.

REFERENCES

[1] Karnik, Neeran, Security in Mobile Agent Systems, *Ph.D. dissertation.* Department of Computer Science and Engineering, University of Minnesota, 1998.

[2] B.H. Tay, A. Ananda, A Survey of Remote Procedure Calls, *Operating system Review*, 24(3), PP 63-79, July 1990

[3] J. Cook, Software Engineering Concerns for Mobile Agent Systems, *paper*, proceeding of Workshop on Software Engineering and Mobility, Ontario, Canada, May 2001.

[4] G. Karjoth, N. Asokan, C. Gulcu , Protecting the Computation result of Free-roaming Agents, *Proceedings of Second International Workshop, Mobile Agent 98, Verlage Lecture Notes in Computer Science*, Vol. 1477, PP 195-207, 1998.

[5] G. Vigna. Mobile Code Technologies, Paradigms and Applications, *PhD thesis*. Politecnico di Milano , Italy, 1997.

[6] A. Fuggetta, G. P. Picco and G Vigna, Understanding Code Mobility, IEEE Transactions on Software Engineering, 24(5): PP 342-61,1998.

[7] G. Karjoth, D. B. Lang, Oshima, *A Security Model for Aglet. IEEE Internet Computing* , 1(4), 1997.

[8] James E. White, Telescript technology: foundation for electronic marketplace, *General Magic White Paper* , General Magic, Inc, 1994.

[9] Ted G. Lewis, *where* client / server Software heading? *IEEE Computer*, PP 49-55, April 1995.

[10] J. Baumann, F. Hohl and N. Radouniklis, Communication Concepts for Mobile Agent System, *paper*, Mobile Agents – First International Workshop, MA'97, Germany, 1997.

[11] J.W. Baek, J.H. Yeo, G. Kim, H.Y. Yeom, Cost e.ective mobile agent planning for distributed information retrieval, in: Proceedings of the International Conference on Distributed Computing Systems (ICDCS), pp. 65–72. April 2001

[12] M. Dikaiakos, M. Kyriakou, and G. Samaras.Performance evaluation of Mobile agent middleware: A hierarchical approach. In Proc. of the Fifth IEEE International Conference on Mobile Agents, LNCS, Atlanta, GA, Springer-Verlag. Dec 2001

[13] L. Ismail and D. Hagimont. A performance evaluation of the mobile agent paradigm. In Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications, pages 306—313, 1999

About the author

Tarig Ahmed is an assistant Professor at Dept. Computer sciences, King Faisal University. He holds Ph.D. degree in computer science from Khartoum University, Sudan. His main interested research areas:Mobile agent system architecture, mobile agent security, distributed systems, database Systems and artificial intelligent systems. Tarig is an Oracle Certified professional (OCP) from Oracle company as application developer 6i. Also, he has used .Net technology, C# , Java languages.

E-mail: Tarig_Harbi71@hotmail.com