Performance Evaluation of AGLETS and JADE Mobile Agent Using Encryption and Decryption Time

Dada E. G., Joseph S. B., and M. K. Mishra

Abstract — **The mobile agent approach is a relatively new concept in the distributed systems environment. The agents migrate from Client to server in a network where the state of the running program is saved, transported to the new host, and are stored, allowing the program to continue from the point where it stopped. In this paper, we evaluate the performance of the JADE and Aglet mobile agents. We developed a simulation program to evaluate the performance of the two mobile agents using the Encryption time, Decryption time and file transfer time. Our findings revealed that there is no significant difference between the performances of these two mobile agents using the parameters mentioned before.**

Index Terms — **Aglets, Decryption, Encryption, JADE, Mobile Agent.**

I. INTRODUCTION

MOBILE agents are autonomous programs that move about the network on behalf of their owners while searching for information or even negotiating with other agents. Mobile agents can also be defined as those agents that possess the characteristics of mobility. This means that the agents have the ability to migrate from one host computer to another. This may seem like a trivial characteristic, but the advantages to mobility are both subtle and important. Mobile agents are also known as programs that are able to migrate in a network in order to optimize their consumption of resources, such as network bandwidth, or to adapt to a changing environment.

A mobile agent migrates from one Host to another Host where the data is sourced. This agent could be a control system (in the simplest case, a thermostat), that reads the source data and then interacts with the system to make adjustments. In this model, the mobile agent interacts with the data collection agent at the source. Moreover, the agent could collect and filter data, and then return to the original host. This type of agent could be useful in situations where full-time connections are not always practically possible (such as satellites in low-earth orbit).

Manuscript received November 18, 2010. Performance evaluation of AGLETS and JADE mobile agent using encryption and decryption time.

Dada E. G. is with the Department of Computer Engineering, University of Maiduguri, Nigeria (e-mail: gbengadada2004@ yahoo.com).

Joseph S. B. is with the Department of Computer Engineering, University of Maiduguri, Nigeria (e-mail: sjbassi74@ yahoo.co.uk).

M. K. Mishra is with the Department of Computer Engineering, University of Maiduguri,Borno State,Nigeria, (Corresponding author phone: +234-8065578635 e-mail: mishrasoft1@gmail.com).

The advent of mobile agents' technology attracts a lot of interest from the fields of distributed systems, information retrieval, electronic commerce and artificial intelligence. The emergence of Java, with its support for mobile code, led to heightened research activity in this area. Java is the language of choice for mobile agent systems such as Concordia, JADE, Odyssey, Aglets, Tracy and Voyager. Java also supports development of mobile agents that are tightly integrated with the Web [1].

Mobile agents have been used in a variety of applications including process control and network monitoring. Network monitoring is an ideal application for mobile agents. An agent is provided details of data collection, and then disbursed into a network. The agent collects data, and either communicates the data back to a central server, or migrates back to itself with its data. Process control is another interesting application. Instead of purely collecting data from remote servers, the agents must also monitor and control the devices to which they're attached. Prior to migrating, the agents can be configured for their particular destination. From this perspective, mobile agents are an interesting deployment method for distributed systems.

II. MOBILE AGENT ARCHITECTURE

The mobile agent architectural pattern introduces the ability for agents to migrate themselves between hosts. The agent architecture includes the mobility element, which allows an agent to migrate from one host to another. An agent can migrate to any host that implements the mobile framework. The mobile agent framework provides a protocol that permits communication between hosts for agent migration. This framework also requires some kind of authentication and security, to avoid a mobile agent framework from becoming a conduit for viruses. Also implicit in the mobile agent framework is a means for discovery. For example, which hosts are available for migration, and what services do they provide? Communication is also implicit, as agents can communicate with one another on a host, or across hosts in preparation for migration. The mobile agent architecture is advantageous as it supports the development of intelligent distributed systems that is dynamic, and whose configuration and loading is defined by the agents themselves (see Figure 1).

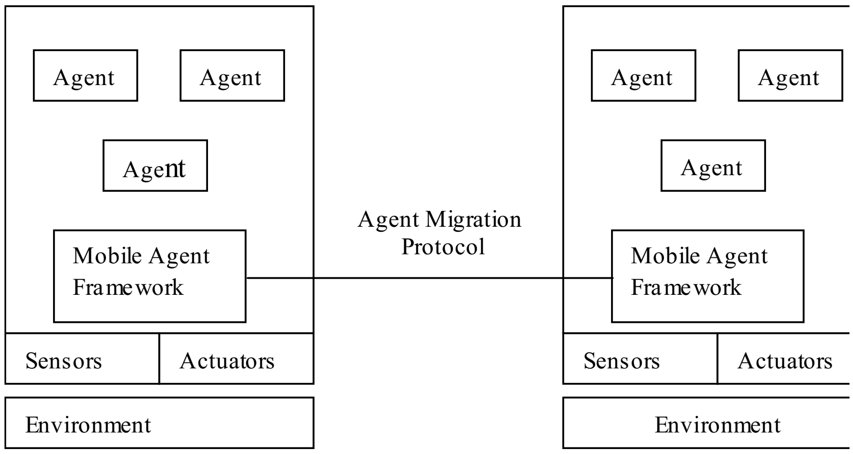

Fig. 1. The mobile agent framework supports agent mobility [5]

A. Technical Obstacles in the Development of Mobile Agents

The Mobile agent paradigm is a promising technology and a new method of communication amongst network nodes. Despite a number of successful mobile agent applications, still there are some barriers preventing this technology from spreading out to a wider range of enterprise and individual users. This is due to many reasons such as, lack of standard in both software and hardware products (e.g. programming languages, protocols and devices). To overcome this, a number of initiatives are underway which may help developers in building their applications based on mobile agent technology as in [3].

Researchers and the developers also find it difficult to define the real concept of mobile agent technology and the tasks that Mobile Agent should perform is another contributing problem. Furthermore, the current infrastructure is not ready to support and integrate with mobile agent technology. Another major concern made by researchers is the security issue, for example, when using MAs whether in E-commerce or M-commerce fields, to act on behalf of their users to handle transactions over the net.

Other unresolved issues include privacy, trust and integrity. Privacy is lost since the agent must have access to the user profile, which may contain sensitive information about the user, and may be shared with other agents in the working environment. In addition, this information may be modified during the transaction (by a hacker for example).

TABLE I
TECHNICAL IMPLICATIONS OF MOBILE AGENTS

Technical Issue	Implication for Mobile Agents
Bandwidth	MAs conserve bandwidth, especially for networks which have low bandwidth capacity (e.g. wireless network). By replacing continuous communication with an agent directly at the point of information generation, the bandwidth use can be reduced. Instead of sending dozens or even hundreds of queries across the network, sending one agent on a single request the agent can manage this process locally at the remote side.
Fault-tolerance	MAs can act or respond on errors that may be encountered within their contexts because of their adaptive and ragged attributes.
Flexibility	MAs can give greater flexibility, because new tasks and codes can be added to the system without the need for a fixed code-base.
Interaction	Mobile agents enable new types of interaction, such as negotiating agents that travel to vendors' sites/servers seeking for the best deal such as comparing prices (e.g. e-commerce application).
Protocols	MAs are able to move (relocate itself) to remote hosts in order to establish "channels" based on proprietary protocols.
Scalability	MAs can carry out their function well (without disruption) when the host system or environment changes in size or volume in order to meet a new user's need.
Self-contained tasks	MAs can carry out tasks which require variable degrees of independence such as, network management, software updates, etc.
Weak coverage	MAs fit perfectly into a disconnected environment where the signal coverage is frequently lost (being disconnected); MAs will then migrate from one node to another when the coverage becomes available.

These issues are trivial but it has to be considered in the mobile agent applications. In other words, MAs need to be protected against hosts, and hosts need to be protected against MAs. In summary, the Table I briefly explains the implications of using MAs for 8 identified significant technical issues.

B. JADE

Java Agent DEvelopment Framework (JADE) is a software framework originally developed by TILAB, Italy and it is totally written in Java. It is an enabling technology, a middleware for the development and run-time execution of peer-to-peer applications which are based on the agents' paradigm. It also simplifies the implementation of multi-agent systems through a middleware that complies with the Foundation for Intelligent Physical Agents (FIPA) specifications. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required.

The conceptual model of JADE dwells mainly on distributed system topology with peer-to-peer networking, and software component architecture with agent paradigm. The network topology affects how the various components are linked together, whereas the component architecture specifies what the components are supposed to expect from one another. The intelligence, initiative, information, resources and control are fully distributed on mobile terminals as well as on computers in the fixed network. Agents otherwise called "peer" evolve dynamically in JADE, appearing and disappearing in the system according to the needs and the requirements of the application environment. Communication between the peers, regardless of whether they are running in the wireless or wired network, is completely symmetric with each peer being able to play both the initiator and the responder role.

The development of JADE according to [4] is based on the following driving principles:

Interoperability: JADE is compliant with the FIPA specifications. As a consequence, JADE agents can interoperate with other agents, provided that they comply with the same standard.

Uniformity and portability: JADE provides a homogeneous set of APIs that are independent from the underlying network and Java version. More in details, the JADE run-time provides the same APIs both for the J2EE, J2SE and J2ME environment. In theory, application developers could decide the Java run-time environment at deploy-time.

Easy to use: The complexity of the middleware is hidden behind a simple and intuitive set of APIs.

Pay-as-you-go philosophy: Programmers do not need to use all the features provided by the middleware. Features that are not used do not require programmers to know anything about them; neither do they add any computational overhead.

Architectural Model: JADE includes both the libraries (i.e. the Java classes) required to develop application agents and the run-time environment that provides the basic services and that must be active on the device before agents can be executed. Each instance of the JADE run-time is called container (since it "contains" agents). The set of all containers is called platform and provides a homogeneous layer that hides to agents (and to application developers also) the complexity and the diversity of the underlying tires (hardware, operating systems, types of network, JVM).

As seen in Figure 2, JADE is compatible with the J2ME CLDC/MIDP1.0 environment. It has already been tested on the fields over the GPRS network with different mobile terminals among which include Nokia 3650, Motorola Accompli008, Siemens SX45, PalmVx, Compaq iPaq, Psion5MX, HP Jornada 560. The JADE run-time memory footprint, in a MIDP1.0 environment, is around 100 KB, but can be further reduced until 50 KB using the ROMizing technique i.e. compiling JADE together with the JVM. JADE is extremely versatile and therefore, not only does it fit the constraints of environments with limited resources, but it has already been integrated into complex architectures such as .NET or J2EE where JADE becomes a service to execute multi-party proactive applications. The limited memory footprint allows installing JADE on all mobile phones provided that they are Java-enabled.

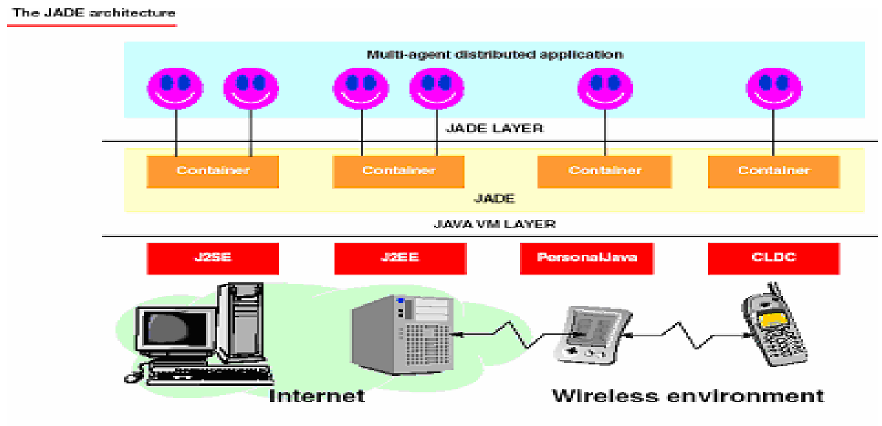

Fig. 2. JADE architectural model (Source: Bellifemine F., JADE a White Paper, Exp Journal)

C. *Aglets*

Aglet is a mobile Java object that visits aglet enabled hosts in a computer network. It is autonomous, since it runs in its own thread of execution after arriving at a host, and reactive, because of its ability to respond to incoming messages [2]. Aglet agent framework was designed by IBM Tokyo in the 1990s. Aglets is based on the Java programming language, as it is well suited for a mobile agents framework. First, the applications are portable to any system (both homogeneous and heterogeneous) that is capable of running a Java Virtual Machine (JVM). Second, a JVM is an ideal platform for migration services. Java supports serialization, which is the aggregation of a Java application's program and data into a single object that is restartable. In this case, the Java application is restarted on a new JVM. Java also provides a secure environment (sandbox) to ensure that a mobile agent framework doesn't become a virus distribution system [5].

The Aglets framework is shown in Figure 3.

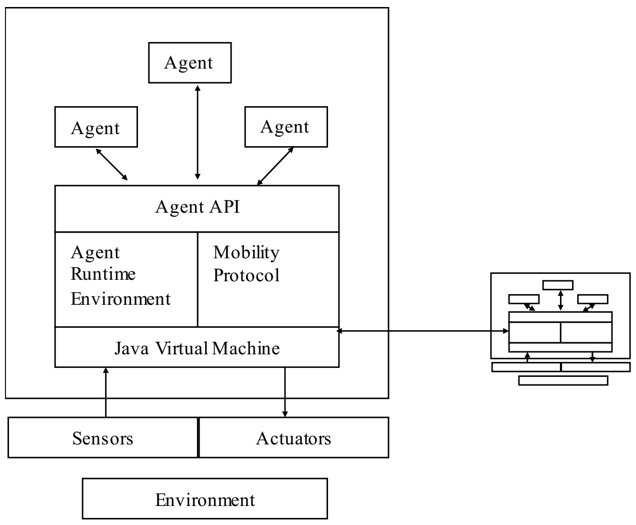

Fig. 3. Aglets mobile agent architecture [5]

At the bottom of the framework is the JVM (the virtual machine that interprets the Java bytecodes). The agent runtime environment and mobility protocol are next. The mobility protocol, called Aglet Transport Protocol (or ATP), provides the means to serialize agents and then transport them to a host previously defined by the agent. The agent API is at the top of the stack, which in usual Java fashion provides a number of API classes that focus on agent operation. Finally, there are the various agents that operate on the framework. The agent API and runtime environment provide a number of services that are central to a mobile agent framework. Some of the more important functions are agent management, communication, and security. Agents must be able to register themselves on a given host to enable communication from outside agents. In order to support communication, security features must be implemented to ensure that the agent has the authority to execute on the framework. Aglets provides a number of necessary characteristics for a mobile agent framework, including mobility, communication, security, and confidentiality. Aglets provide weak migration, in that the agents can only migrate at arbitrary points within the code (such as with the dispatch method).

III. Discussion

We developed an application for the implementation of two mobile agent systems on win32 platform that is, windows XP SP2. Five dummy files whose size ranges from 500kb to 1mb were created on the client system. The results of the comparison encryption and decryption time between Aglets and JADE can be view from the server. The time is measured in milliseconds while the size of the files is measured in bytes. From the experiments performed, it was observed that there is difference in encryption and decryption time between Aglets and JADE with the latter giving a better performance than Aglets in all test cases. The time difference was not much, it was also observed that the time taken to encrypt, decrypt and transfer files increases as the file sizes increases.

IV. Analysis of Results

The Table II is a comparison between the encryption time of each file from 500kb – 1mb for JADE and Aglets (see Figure 4).

TABLE II
ENCRYPTION TIME COMPARISON BETWEEN JADE AND AGELETS

	Encryption Time (ms)	
Size (kb)	JADE	Aglet
500	760	765
600	1681	1686
700	1700	1705
800	1761	1766
900	1971	1977
1000	2052	2058

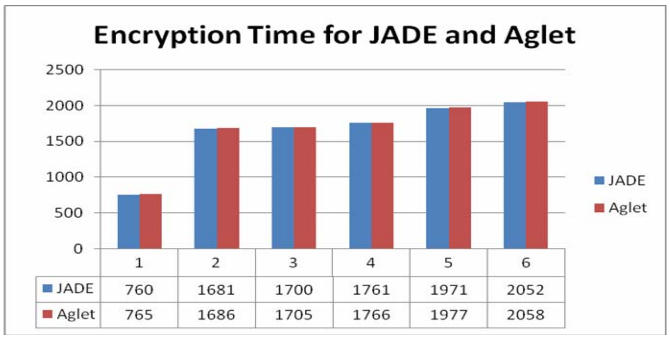

Fig. 4. Graph showing encryption time comparison between JADE and Aglets

The Table III is a comparison between the decryption time of each file from 500kb – 1mb for JADE and Aglets (see Figure 5).

The Table IV shows the comparison of JADE and Aglet in terms of the time it takes to send files from one computer system to another one (see Figure 6).

It was observed from the Table IV that the rate at which JADE transfers the encrypted files is to some extent faster than that of Aglets.

TABLE III
DECRYPTION TIME COMPARISON BETWEEN JADE AND AGELETS

	Decryption Time (ms)	
Size (kb)	JADE	Aglet
500	302	300
600	455	451
700	510	504
800	612	1766
900	667	661
1000	730	731

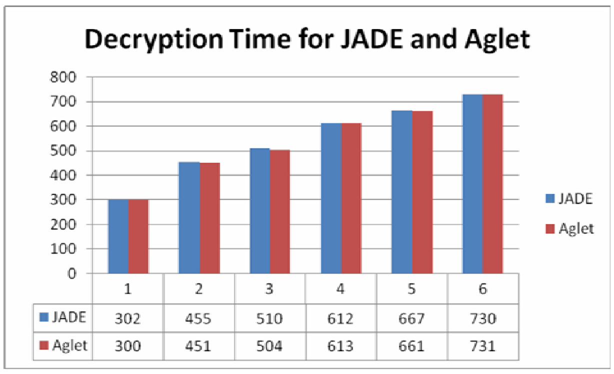

Fig. 5. Graph showing decryption time comparison between JADE and Aglets

TABLE IV
FILE TRANSMISSION TIME COMPARISON BETWEEN JADE AND AGELETS

	Time taken to send files from one system to another (ms)	
Size (kb)	JADE	Aglet
500	169,337,778,715	169,337,778,713
600	169,337,778,775	169,337,778,773
700	169,337,779,790	169,337,779,788
800	169,337,780,037	169,337,780,034
900	169,337,780,318	169,337,780,315
1000	169,337,780,500	169,337,780,497

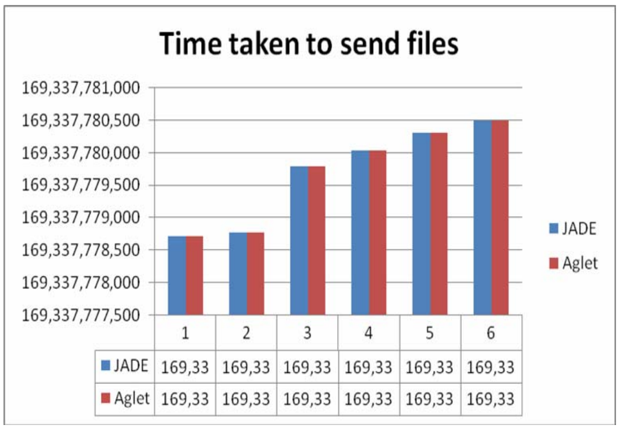

Fig. 6. Graph showing file transmission time comparison between JADE and Aglets

V. CONCLUSION

The results obtained showed that there is only some slight difference in performance between JADE and Aglet in terms of Encryption time, Decryption time and file transfer. The little differences could be due to the fact that all information exchanged by JADE complies with FIPA specification and hence include only the information required by the transport layer unlike Aglet that exchanges all data. Another factor is that JADE support skeletons that are implemented as abstract classes that relief the programmer the burden of solving synchronization, timeouts and other challenges. Also JADE uses the asynchronous method of messaging, which puts the Agent Communication Language (ACL) into consideration and supports multiple agent execution and interaction. This makes it more preferred in multi agent distributed environments. Aglets use API for transfer of agent and RMI for exchange of messages which is not in line with FIPA regulations that JADE is using. Our future work will focus on using memory utilization and fault tolerance to test the performance of mobile agents across different platform.

REFERENCES

[1] Wong, D., Paciorek, N. and Moore D., "Java-based mobile agents," Communications of the ACM 42(3), 1999, pp. 92-105.

[2] Lange, D. B. and Oshima, M., "Programming and deploying Java mobile agents with aglets," Addison-Wesley, 1998.

[3] FIPA, "Foundation for intelligent physical agents", 2004, URL: http://www.fipa.org/

[4] Bellifemine, F., "JADE – a white paper. EXP. 3.3", Retrieved October 1, 2010 from http://exp.telecomitalialab.com, , 2003

[5] Jones Tim M., "Artificial intelligence: A System Approach.," Infinity Science Press LLL, 2008.