ISSN: 2141 – 3290
www.wojast.com

INFORMATION RETRIEVAL USING JADE MOBILE AGENT SYSTEM

EFFANGA[1], E.N., ASUQUO[2], D. E. AND WILLIAMS[3] , E. E.

[1]Department of Computer Science, University of Ibadan, Ibadan, Nigeria
[2]Department of Computer Science, University of Uyo, Uyo, Nigeria
[3]Department of Mathematics/Statistics & Computer Science University of Calabar, Calabar, Nigeria
[2]Corresponding author: E-mail: danasu4all@yahoo.com

ABSTRACT: With the introduction of the Internet, computer systems and users are becoming increasingly interactive and the internet in particular, with huge amount of data becomes a viable platform for interactive computing. Unfortunately, access to desired information in the internet is often times difficult due to poor human-computer interaction. One of the best means of getting services or finding information of a particular interest on a network requires the combine use of two software: mobile agents and an interface, to connect users and resources in a transparent, open, and scalable way. We use Java Agent Development Framework - JADE mobile agent system to develop a web-based user interface that enables information retrieval from a database. The implementation featured the use of a browser acting as a client, and a server connected to the agent platform. The developed interface enables users' access and retrieves information stored in the server in a more efficient manner, move processing from the client to the server, and boost the system's performance in terms of network latency, bandwidth and memory usage.

INTRODUCTION

The Internet has rapidly become the medium of choice by computer users for utilizing information technology to achieve individual, business and organizational objectives. As Internet and World Wide Web (WWW) usage is further integrated into the professional and personal lives of end users, ensuring that the needed information can be found, gathered, and retrieved efficiently. This, according to Kotz and Gray (1999), is because the amount of available on-line information expands equally with the Internet. There is therefore need for a tool that will enhance efficient access and retrieval of information as well as a format for displaying the retrieved information to users. This solution is provided by the use of mobile agents and an interface (Williams *et al*, 2007). In a nutshell, mobile agents (MAs) are autonomous programs that move about the network on behalf of their owners while searching for information or even negotiating with other agents. They are processes (i.e., executing programs) that can migrate from one machine of a system to another machine (usually in the same system) in order to satisfy requests made by their clients (Johansen *et al*, 1995). According to Lange and Oshima (1998) and Wong *et al* (1999), Java has become a language for mobile coding and integration with the Web. The interface accepts input from users and provides a means of displaying retrieved information to users. But the real concern, apart from movement of agents, would also be the format of displaying to users, information retrieved from the server.

The agent's Graphical User Interface (GUI) lacks this functionality and tends to use up more system resources like memory. Considering the use of mobile agents in web application,

Asuquo *et al* (2008) opine that a more robust user interface is needed to process information on the web as well as improve the mode of displaying retrieved information. In this paper, we present a software model for mobile agent deployment. To improve interactions between users and the system, we develop a web-based user interface that makes use of mobile agents running on Hypertext Transfer Protocol (HTTP) and retrieving desired information from a database. This shifts most of the processing to the server-side thereby conserving the memory usage of the system for download time and installation of Java Runtime Environment (JRE).

MOBILE AGENT PARADIGM

Mobile agents have been developed as an extension to and replacement of the client-server model (Gray, 1995). A limitation of the client-server model is that the client is limited to the operations provided at the server. So, if a client needs a service that a particular server does not provide, the client must find a server that can satisfy the request by sending out messages to all servers. This clearly is an inefficient use of network bandwidth. Also, this severely limits network scalability since managing and updating these servers would prove prohibitive (Gray *et al*, 2001). Traditionally, distributive applications have relied on the client-server paradigm, in which client and server processes communicate either through message passing or Remote Procedural Call (RPC). This communications model is usually synchronous, i.e., the client suspends itself after sending a request to the server, waiting for the result of the call. An alternative architecture called Remote Evaluation (REV) was proposed by Stamos and Gifford (1990) in Karnik and Tripathi (1998). In REV, the client, instead of remote procedure, sends its own procedure code to a server, and request the server to execute it and return the result.

The mobile agent paradigm has evolved from these antecedents. Figure 1 shows how it differs from RPC and REV. In RPC, data is transmitted between the client and server in both directions. In REV, code is sent from the client to the server, and data is returned. In contrast, a mobile agent is a program (encapsulating code, data and context) sent by a client to a server. Unlike a procedural call, it does not have to return its result to the client. It could migrate back to the client if appropriate. It thus has more autonomy than a simple procedure call.

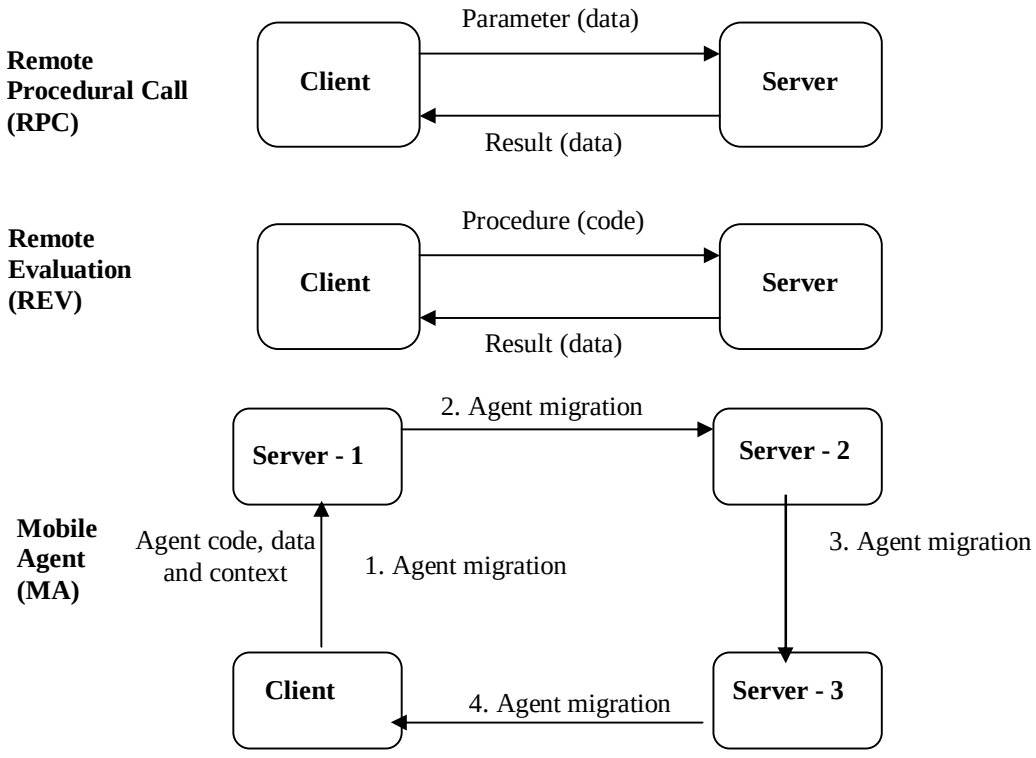

Figure 1: Evolution of the Mobile Agent Paradigm

Mobile agents are programs that can migrate from host to host in a network, at times and to places of their own choice. The mobile agent approach shown in figure 2 is a relatively new paradigm in the distributed systems environment (Kotz and Gray, 1999). 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 restored, allowing the program to continue where it left off. Mobile agent systems differ from process-migration systems in that the agents move when they choose, typically through a "jump" or "go" statement, whereas in a process-migration system the system decides when and where to move the running process (typically to balance CPU load). Mobile agents differ from "applets", which are programs downloaded as the result of a user action, then executed from beginning to end on one host.

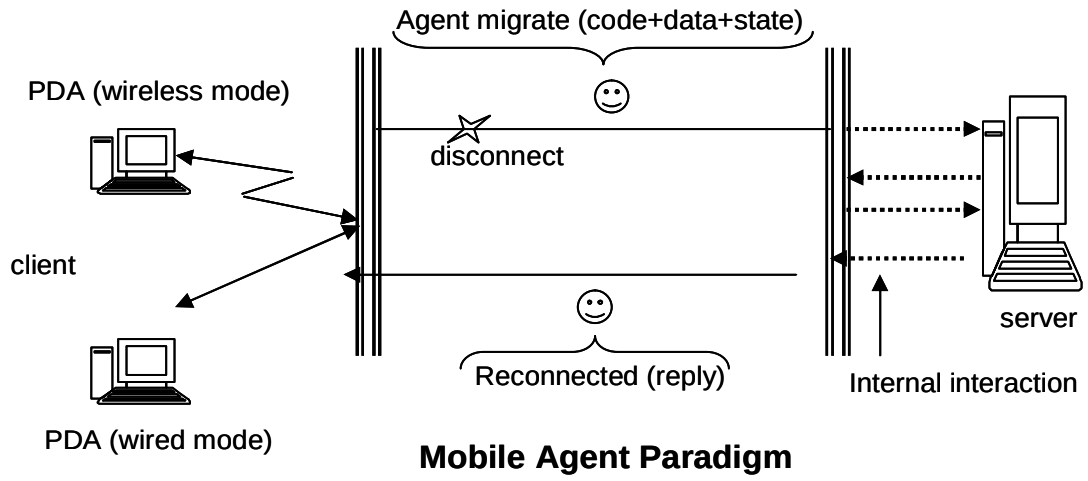

Figure 2: Mobile Agent Paradigm

Several types of agent exist. These agents may either be static or mobile in nature and built based on their purpose. Static agents tend to be resource-intensive and possess some form of intelligence or decision support, the focus is cooperation whereby multiple static agents combine to perform a task. Mobile agents on the other hand are lightweight and compact and are designed to migrate from host to host. The rationale behind such agents is the ability to perform tasks at source via executing the mobile-code, thus allowing minimal bandwidth usage in resource-intensive computations and potential performance balancing.

Mobile agents are one form of mobile code. In its simplest form, the concept of mobile code involves the dynamic installation of code on a remote host. In Web applications, applets and servlets are a common form of mobile code. The mobile code concept also appears in "remote evaluation" systems (Stamos and Gifford, 1990), which extend the notion of remote procedure call to transport the procedure to the server along with the call. Many researchers extend the mobile-code concept to "mobile objects," in which an object (code and data) are moved from one host to another. The mobile-agent abstraction or approach extends this concept further by moving code, data, and state (thread) from one host to another as well. Mobile agents run in one location, move with their state to another host, and continue execution at that host. While mobile code and mobile objects are normally moved by an external entity, mobile agents usually have migration autonomy.

Mobile agent technology has been employed in many areas from network management tasks to information management. The use of mobile agents in the wireless environment means that the application should support disconnected mode. For example, when the mobile agent is sent

from a Personal Digital Assistant (PDA) to the network, the PDA could be disconnected and later when the connection is re-established the mobile agent with the desired results can migrate back to the source (Wang *et al*, 2003). This feature gives the mobile agent a significant advantage over other communication paradigms (e.g. Client-Server, Remote Procedure Call). Further developments have been done using mobile agent technology on some application domains such as data warehouses, software updates, and information management tasks such as searching for information, information filtering, information monitoring and wireless network tasks. As recorded in Williams *et al* (2007), Bigus and Bigus (1998) and Lange and Oshima (1999) observed that the use of MAs leads to increase performance by reducing network load, overcoming network latency, encapsulating protocols in heterogeneously distributed systems and making them robust and fault tolerant, executing asynchronously and synchronously, operating asynchronously and autonomously.

METHODOLOGY

The Web's continuing success is attributed to the simplicity with which it provides users an ability to locate and retrieve dispersed information from the Internet. A major part of this success is due to the effectiveness of the protocol, HTTP through the Web's architecture and format of storing/referencing information. Computers have become interactive. Preece (1994) asserts that an interface is a means of improving interaction between the user and the computer when resources are too numerous to handle.

We adopt the JADE architectural model in the methodology, develop and implement the mobile agent code with Java Programming Language and develop a web-based GUI for the JADE mobile agents to display the status of the platform, consisting of retrieved information stored in a database system. The web-based interfaces are developed using different technology such as Java applets, servlets and HTML. In accordance with Aneiba and Rees (2004) requirements, our mobile agent software includes: mobile agent program (code), mobile agent platform (executable environment), and GUI-based program. The developed agent is robust and capable of performing operations such as Creation, Cloning, Dispatching or Migration, Retraction, Activation, Deactivation and Disposal (Lange and Oshima, 1999).

a) **Creation** is the first period in the mobile agent life cycle. For example, when a system requests a service to be done by the mobile agent, the system must create a mobile agent instance in the first place before any work can be done. During, this creation stage the mobile agent will be equipped with the desired parameters in order to achieve its goal.
b) **Cloning** basically creates a copy of the original mobile agent object. This operation is used when the need for another agent (with the same features and properties) to do the same or other job in conjunction with the original one.
c) **The dispatching or migrating** is another main function in the mobile agent life cycle. This function is responsible for moving the agent from one node to another within the network environment by specifying the destination address e.g. Unified Resource Locator (URL) to the agent. There are two main types of migration: strong and weak. Strong migration means that the mobile agent can carry itself, its data and its state while weak migration means that mobile agent can carry only itself and its data (e.g. mobile object).
d) **The retraction function** is done where the agent's source node requires its agent to be returned to the original host or node.
e) **Activation and deactivation** are operations done when the mobile agent is required to start or to stop only within certain time of its lifetime. Finally, the dispose operation is done where the agent life comes to the end.

In order to program and run the mobile agent application, mobile agent platform must be implemented and exist. According to Marques *et al* (2001), MAs' special characteristics and

behaviors require the host to know how to speak with the incoming agents and with their applications. The host will then provide the execution needs to those arriving agents in order to achieve their goals. Mobile agent platforms should provide the following important requirements: platform independence, authentication, secure execution, dynamic class loading, network connectivity, and resources control (Suri *et al*, 2000). It is observed that both the amount of information available on the internet and the number and diversity of its users, are growing rapidly. This diverse population of user will not settle for a uniform interface to the information, but will demand personalized presentations and access methods. This personalization will range from different presentation formats to complex techniques for searching, filtering and organizing the vast quantities of information. The interface can be a text or GUI-based program at the user-end. In this the user may need to set up some agent's parameters before leaving the client ground to land at the server.

JADE ARCHITECTURAL MODEL

Java Agent DEvelopment Framework (JADE) is a software 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. The conceptual model of JADE deals mainly on distributed system topology with peer-to-peer networking, and software component architecture with agent paradigm. 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 Bellifemine *et al* (2003) is based on the principles of interoperability with Foundation for Intelligent Physical Agents (FIPA) specifications (FIPA, 2004), uniformity and portability of Application Programming Interfaces (APIs) with different Java run-time environments, ease of use of the APIs by hiding the complexities of the middleware from the programmer and reduction of computational overhead.

JADE architectural model includes both the libraries required to develop application agents and the run-time environment that provides the basic services which must be active on the device before agents can be executed. These services include agent identification and agent communication. 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, Java Virtual Machine- 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 (Sun, 2005). The limited memory footprint allows installing JADE on all mobile phones provided that they are Java-enabled. Figure 3 shows the JADE architectural model.

JADE comes with a platform and container manager and offers a GUI for managing the platform and containers. From this interface the administrator can easily manage the available containers (locally) in the platform as well as the ones distributed to other systems. The functionality of these interfaces whether it is agent-based or web-based, requires a protocol at application level. But, apart from protocols, the places of deployment of these GUIs (either within or outside a browser) have some impacts on the performance of the system. The GUI is usually at the user-end acting as a sub-software system (or a tool) that provides controls and management of mobile agents. Inherent in this tool are features required for access of all agents' services such as agent transfer or communication. The GUI is also for displaying the status of the platform in order to know the agent at work. Although JADE server supplies basic

services to agents, the mobile agent creator or GUI is not robust when considering services for web-based applications like information retrieval. This work develops a robust web-based GUI for the JADE mobile agents to display the status of the platform.

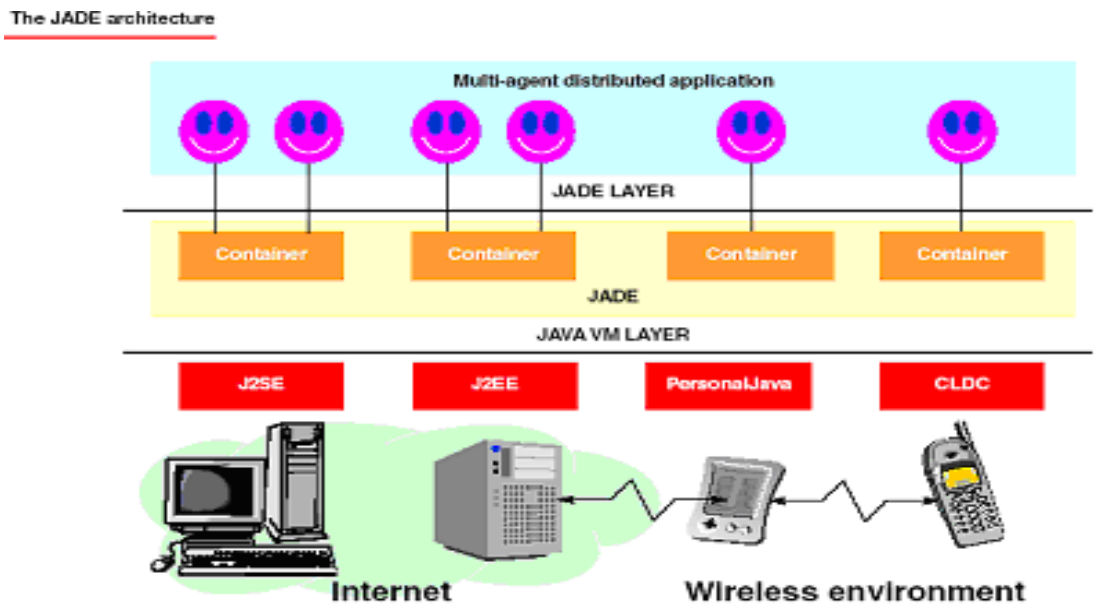

Figure 3: JADE Architectural Model

RESULTS AND DISCUSSION

The system design shown in figure 4 comprises three modules, which are:

a) A web-based user interface
b) The JADE platform
c) The database system

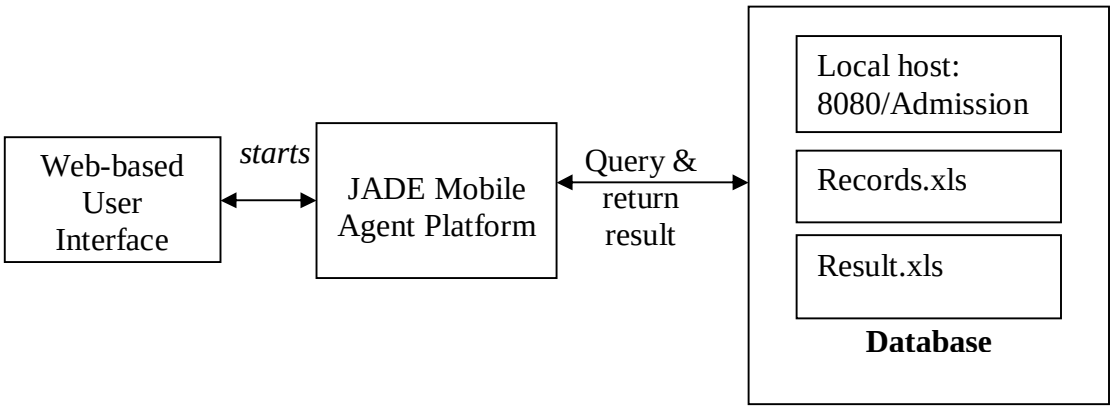

Figure 4: System design for information retrieval using JADE Mobile Agent Platform

The web-based user interface module comprises a web browser, application server and links that provide connectivity between them (Figure 5). Generally, the web-based user interface's architecture provides the user with a medium for easier management of agents within a standalone system and access to the complete set of JADE mobile agent system operations. The web browser is an application that is used for fetching and displaying of web pages. It allows

users to interconnect to the application server, using an addressing standard called URL. Usually, the address is broken down by the web browser into 3 major parts as follows:

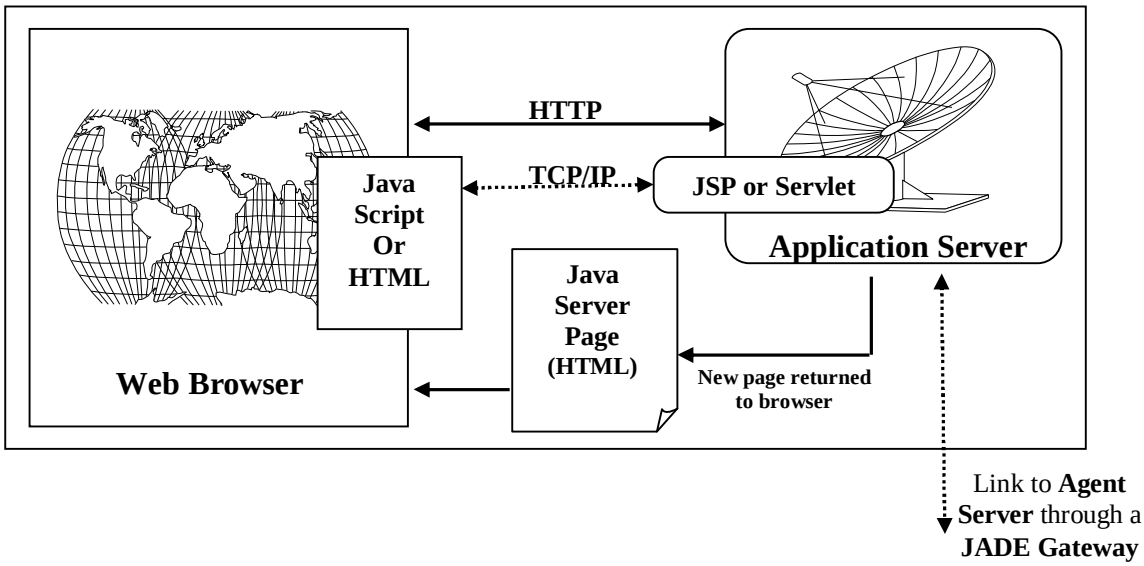

Figure 5: Architecture of web-based user interface

a) **HTTP**: This is a protocol used by the browser to make request for information from the server. From application point of view, HTTP is the first addressable form of the URL.
b) **The Server Name:** The application will be implemented in a standalone system; hence the server name "*localhost*" is used. The client/server model would still be employed, with served web pages residing in the local server. In the case of a remote server, the name will require the specification of the Internet Protocol (IP) address of such machine.
c) **The file name:** This is a named file found in the server.

The application server handles all data processing when requests are made to it via the browser. Pages are sent back and displayed on the browser in form of Hypertext Markup language (HTML) or Java script but reside on the server in the form of Java Server Page or Servlet. Access to the page would require the use of numbered ports, one for each available service. The combine usage of HTML forms, Java Applets and Servlets in the design is aimed at taking care of the stateless nature of the HTTP protocol and any breach in communication between the browser and the agent server. The robust, routable and efficient Transmission Control Protocol/Internet Protocol (TCP/IP) is used in the design for the purpose of maintaining communication links and data transfer between the server and the browser.

The JADE platform module acts as a middleware that facilitates the development of multi-agent systems as well as those accessible to web-based applications. In Figure 6, the JADE platform module comprises the host container and the database system.

The host container is the running instance of the JADE runtime environment with several agents contained inside it. For this research, the agents in the container include:

- Effanga: the search agent
- Gateway Agent: responsible for managing and controlling of agent services
- Agent Management System (AMS): provides the naming service and ensures that each agent in the platform has a unique name
- Directory Facilitator (DF): assist an agent to locate another agent that provides the services its requires (otherwise known as Yellow Pages service)

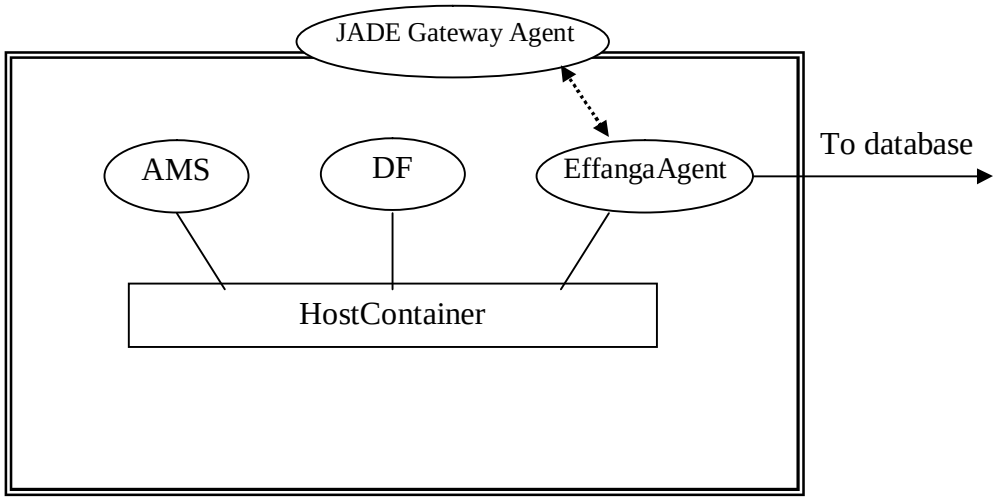

Figure 6: Web-accessible JADE platform architecture

The host container also constitutes the main class that hosts the mobile agent for JADE. The class contains the following methods which are specialized to perform certain functions:

public Admin()
public Search()
public Result()
public void init
public void preprocess
public void prerender
public void destroy

The runtime environment is required to remain active in the platform likewise all other containers in it, so as to enable the execution of the agents. The database system is designed to consist of two different database file: *internal and external*. These databases differ from each other in terms of their location in the host machine. While the internal database is designed for use (updates and retrievals) on the browser, the external one requires (or uses) a Microsoft Excel application program. As shown in Figure 7, the system architecture is designed such that the JADE agent interact with software resources such as files storing information, or databases only during file retrieval (or search) process. The insertion of data and update are meant to be carried out by users, which are usually human operators. In any case, results are displayed after each process.

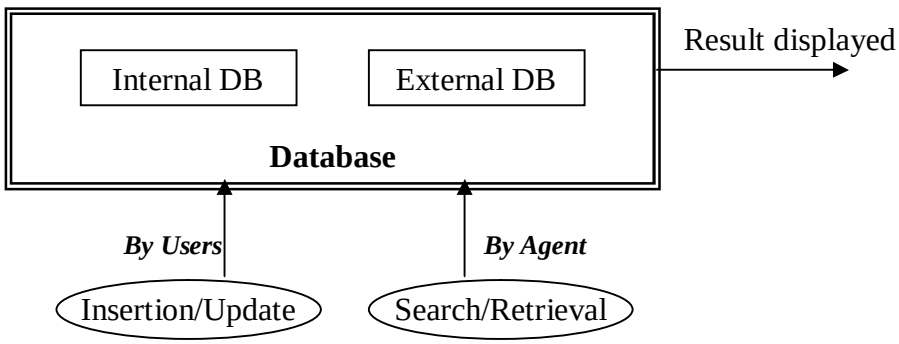

Figure 7: Database system (DB) architecture

The database (the internal one) is an open source Apache Derby, with 100% Java technology Relational Database Management System (RDMS) that comes embedded in the application server. For an excellent integration with the database and guarantee application scalability,

Object Relational Mapping (ORM) was chosen. ORM is a programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages. ORM helps the application communicate with the database by converting Java Entity classes into database specific Structured Query Language (SQL). This in turn, helps to ensure the portability of the application hence it is not tied to a particular RDBMS.

The software components required for the implementation of the web-based user interface are basically default application server, JavaDB and JADE platform. The JADE startup, platform GUI, Admin web-based screen, and searched and retrieved information screens are as shown in Figures 8, 9, 10, and 11 respectively. A blank web page comes up on clicking Internet Explorer via start menu. The blank web page results from an empty URL, which has "about: blank" address. To get the designed user interface, the URL address required would be "http://localhost:8080/Effanga". The Admin web-based screen shown in Figure 10 is where the administrator defines the course schedules by making inputs as course name, start and end dates and then submits the form for server side processing. One characteristic feature of this application is that all processing are carried out on the application server. This occurs as a result of the use of HTTP and TCP/IP. An internal database file exist, where all data processed at the server side eventually end up. Also, there is provision for a menu that allows the administrator to import external data into the application. Through the "Window Task Manager", it is observed that the application uses less memory requirement compared to processing carried out on the client side.

The search web-based screen forms the interface part of the application where the user makes a search from inputs that had already been sent to the server. The interface as shown in Figure 11 consists of an input box for information retrieval and a search button, which is clicked to submit the form for processing. The search generally is handled by JADE mobile agent called 'Effanga'. The user causes an event (insertion or search) and it generates a POST message to the servlet and the *sendmessage* action is invoked.

The proposed approach is based on the creation of a dedicated agent (called GatewayAgent), acting as gateway between the JSP and the JADE world. The action creates a new *BlackBoard* object which will be the message channel between the *GatewayAgent* and the servlet. The GatewayAgent extracts what the request is. (For this research, the message was for information search or retrievals). After that it sends the request to the search agent (Effanga) who now searches the database. The search is based on the agent codes (classes). When the needed information is found, the search agent sends the reply to the GatewayAgent, who packs the reply and sends it via BlackBoard to the servlet who then forwards the result to the browser. The result of the search can be seen in Figure 12.

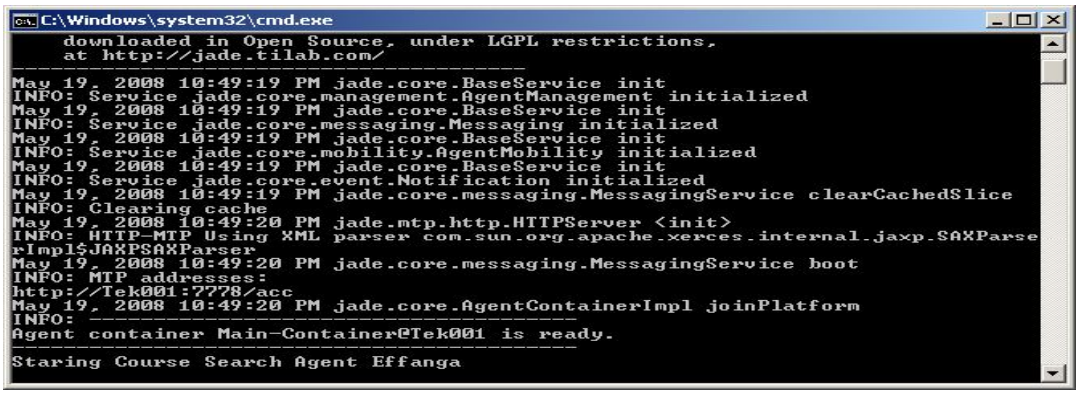

Figure 8: JADE startup

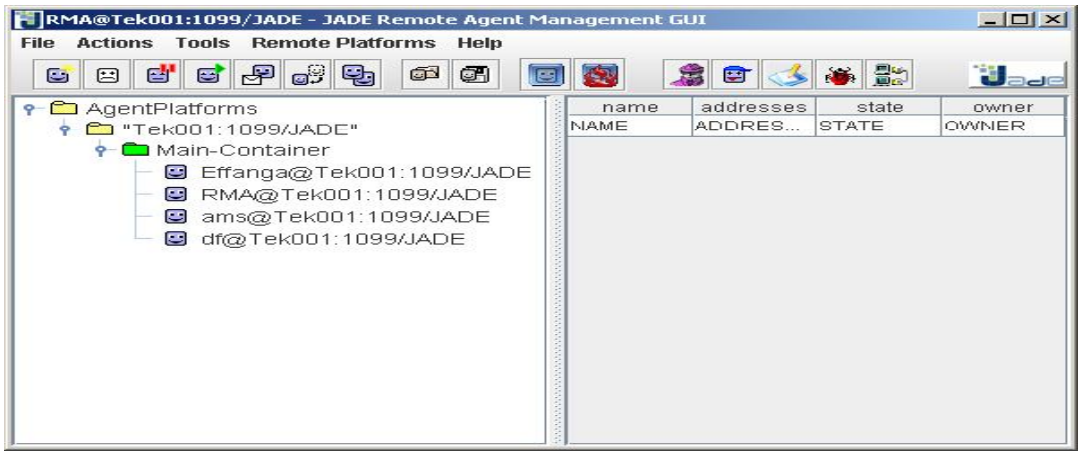

Figure 9: JADE Platform GUI

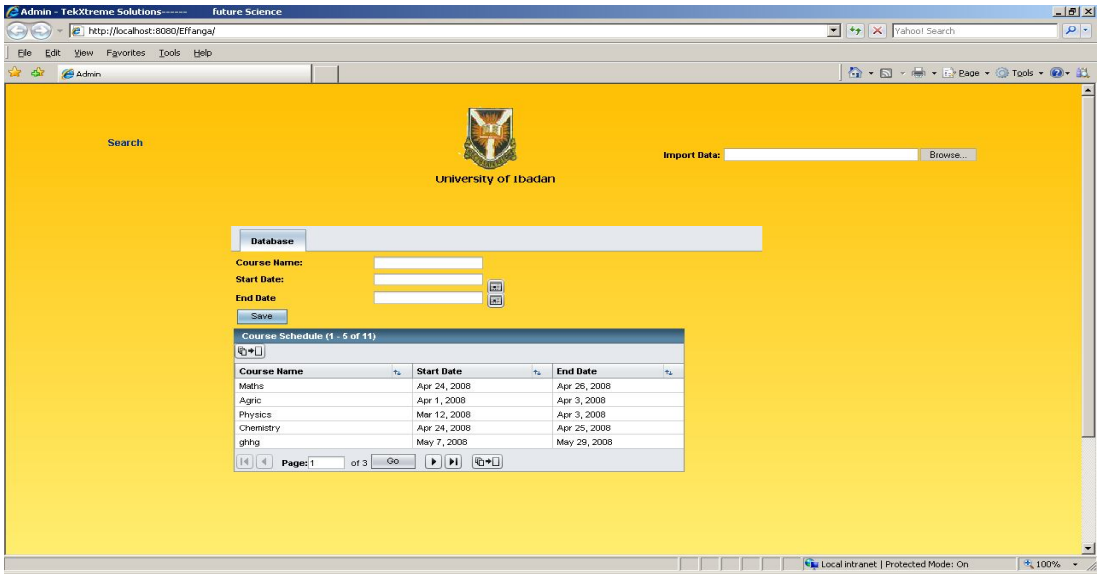

Figure 10: Admin web-based screen

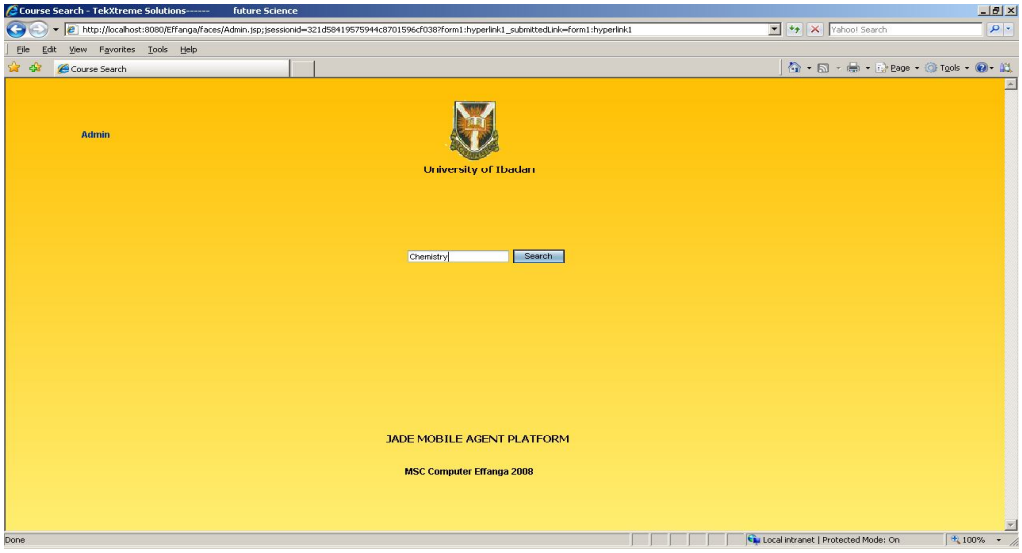

Figure 11: Search web-based screen

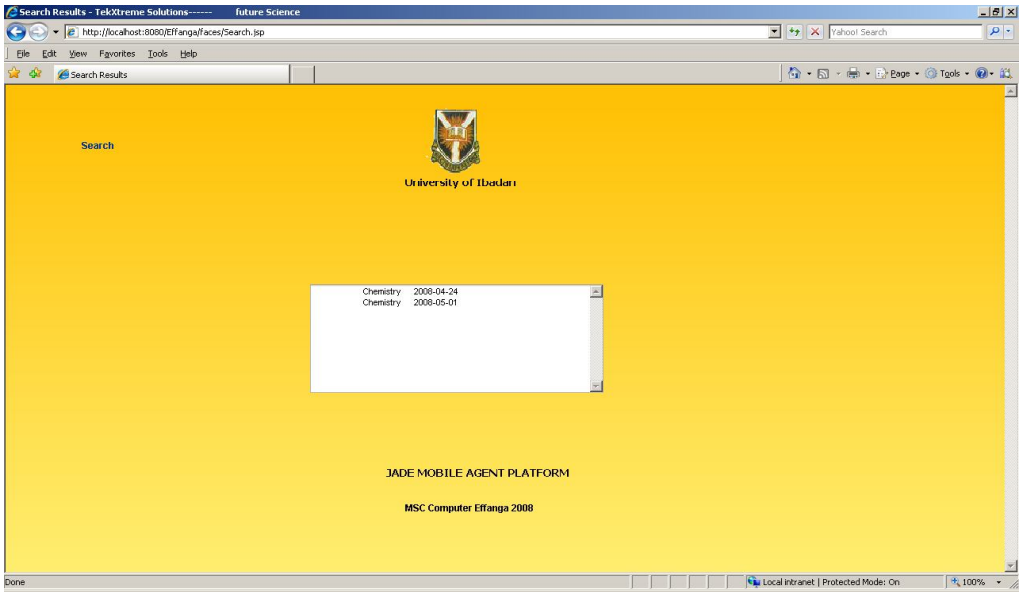

Figure 12: Result from information retrieval

CONCLUSION

In this research, we developed a web-based user interface for information retrieval using the JADE mobile agent system. We also used the JSP and JDK development tools with HTTP and TCP/IP technologies connecting it to the JADE platform via the application server. The implementation was carried out on a standalone system (local host), which acted both as a client and server. For information retrieval, the interface was most preferred as compared to others like agent's own GUI or Java Applets-based GUI. Little processing was carried out by the client while a major part moved to the server.

With the web-based user interface, information can be retrieved easily by the client with less system requirement like memory usage. Moreover, the deployment of JADE mobile agent makes for the easy development of the application due to its open source-code, interoperability with other agents, availability and ease-to-use.

We recommend that an extension of this work should consider technical issues affecting mobile agents' deployment such as security, privacy, trust and integrity. Performance and functional limitations resulting from security in its executing environments, virus scanning and epidemic control mechanisms, transmission efficiency, are also important considerations.

REFERENCES

Aneiba, A. and Rees, S. J. (2004). Mobile Agents Technology and Mobility. http://www.soc.staffs.ac.uk/aa11/Mobile_Agents_Tech_and_mobility.pdf

Asuquo, D. E., Williams, E. E., Oluwade, B. A. and Bassey, P. C. (2008). Educational Websites Usability Evaluation. *Journal of Engineering and Applied Sciences*, 3(7): 574-582.

Bellifemine, F., Caire, G., Poggi, A. and Rimassa, G. (2003). JADE- a white paper. *EXP in search of innovation*, (3) 3, 6-19.

Bigus, P. and Bigus, J. (1998). Constructing Intelligent Agents with Java. John Wiley and Sons Ltd, England. ISBN: 047-19135-3.

FIPA, (2004). Foundation for Intelligent Physical Agents. http://www.fipa.org/

Gray, R. S. (1995). "Agent Tcl: A transportable agent system". *Proceedings of the CIKM workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management (CIKM 95)*, Baltimore, Maryland.

Gray, R., Kotz, D. and Peterson, R.A. (2001). Mobile-agent versus client/server performance: scalability in an information-retrieval task. Technical Report, TR2001-386. ftp://ftp.cs.dartmouth.edu/TR/TR2001-386.ps.Z

Johansen, D., Renesse, R. v., Schneider, F. B. (1995). "Operating system support for mobile agents", *Proc. of the 5th IEEE HOTOS Workshop*, Orcas Island, USA.

Karnik, N. M. and Tripathi, A. R. (1998). Design Issues in Mobile Agent Programming Systems. *IEEE Concurrency*, (6) 6, 52–61.

Kotz, D. and Gray, R. S. (1999). "Mobile Agents and the Future of the Internet", *ACM Operating Systems Review*, (33) 3, 7-13.

Lange, D., and Oshima, M. (1999). Seven good reasons for Mobile Agents. *Communications of the ACM*, No. 43, (3), 88–99. ISSN: 0001-0782.

Marques, P., Simões, P., Silva, L., Boavida, F. and Silva, J. (2001). Providing Applications with Mobile Agent Technology, Open Architectures and Network Programming *IEEE*, 129-136.

Preece, J. (1994). Human-Computer Interaction; Addison-Wesley, New York. ISBN 0201627698.

Stamos, J. W. and Gifford, D. K. (1990). "Remote Evaluation", ACM Transaction on Programming Languages and Systems, (12) 4.

Sun (2005). Sun Microsystems Inc. Java 2 Platform, Enterprise Edition (J2EE), 1994-2005. http://java.sun.com/j2ee/.

Suri, N., Carvalho, N., Bradshaw, R. and Bradshaw, J. M. (2000). Small Mobile Agent Platforms. http://autonomousagents.org/ubiquitousagents/papers/papers/32.pdf

Wang, A., Sørensen, C. and Indal, E. (2003). A Mobile Agent Architecture for Heterogeneous Devices. http://www.idi.ntnu.no/grupper/su/publ/alfw/woc2003-mobile-agent.pdf

Williams E. E., Asuquo, D. E. and Effanga, E. N. (2007). A Survey of Mobile Agent Systems and Paradigm, *International Journal of Natural and Applied Sciences (IJNAS)*, (2) 2, 97-102.

Wong, D., Paciorek, N. and Moore, D. (1999). Java-based Mobile Agents. *Communications of the ACM* , (42), 3, 92-105.