Issues in Informing Science and Information Technology Volume 3, 2006

Remote Method Invocation and Mobil Agent: A Comparative Analysis

G.A. Aderounmu
Obafemi Awolowo University
Ile-Ife, Nigeria

gaderoun@oauife.edu.ng

B.O. Oyatokun
Olabisi Onabanjo
University, Ago-Iwoye,
Nigeria

M.O. Adigun
University of Zululand, Kwadlangezwa,
Republic of South Africa

madigun@pan.uzulu.ac.za

Abstract

This paper presents a comparative analysis of Remote Method Invocation (RMI) and Mobile Agent (MA) paradigm used to implement the information storage and retrieval system in a distributed computing environment.

Simulation program was developed to measure the performance of MA and RMI using object oriented programming language (the following parameters: search time, fault tolerance and invocation cost. We used search time, fault tolerance and invocation cost as performance parameters in this research work.

Experimental results showed that Mobile Agent paradigm offers a superior performance compared to RMI paradigm, offers fast computational speed; procure lower invocation cost by making local invocations instead of remote invocations over the network, thereby reducing network bandwidth. Finally MA has a better fault tolerance than the RMI. With a probability of failure pr = 0.1, mobile agent degrades gracefully.

Keywords: Mobile Agent, RMI, remote object, query, invocation.

Introduction

In the past few years, there has been a growing interest in using mobile agent technology for the next generation enterprises. This is because, the next generation enterprises will require the ability to efficiently retrieve, organize, manage and leverage information and knowledge from widely dispersed sources within and without virtual organization (Seng, 1999). Mobile agents have been specifically proposed to automate the task of retrieving; organizing and filtering information located in widely dispersed sites (Dale and

DeRoure, 1997). Although agent technology is a branch of artificial intelligence, it is believed that mobile agent technology can be employed to build distributed information management tools. There are various reasons for using mobile agent (Lange & Oshima, 1999). This work presents the performance benefits of mobile agent for information storage and retrieval in a distributed computing environment over the exiting client/server (Remote Method Invocation) method.

Several works have been done in developing mobile agent system for information storage and retrieval, but not much has since been done in the area of comparison of the different approaches for distributed information management.

In (Aderounmu, 2004), the use of mobile agent in the control and transfer of data in distributed computing environment was presented against the traditional client-server computing using RPC (Remote Procedure Call). The author explores bandwidth utilization, percentage of denial of service, and network overload with retransmission to show a proof of superior scheme provided by mobile agent as against the client server computing.

Mobile Agent in Distributed Information Retrieval (Brewington, Gray, Moizumi, Kotz, Cybenko, & Rus, 1999) analyzes base performance in terms of response time (the interval between when a request is sent and the time the result is received). The authors made attempt to prove that mobile agent incurs significantly more overhead than RPC due to inter-agent communication and migration when the number of queries and relevant document per query are few; however MA begins to perform excellently as the number of queries and relevant document per query increase.

In (Kunal, 2003), the author simulated the basic behaviour of mobile agent by extending a common general-purpose simulator and a comparison of MA and RPC was carried out using the number of server required to visit to complete a task, the number of records and the number of interaction with agent sizes. Also Ismail and Hagimont, (1999) evaluated the performance of MA paradigm using cost of basic mechanism in the implementation of MA and RPC as the performance metric. Other related works include: Aderounmu, Adagunodo, Ogwu, and Akinde ,2000; Aderounmu, Adekiigbe, and Iyilade, 2004; Ajayi, Aderounmu, and Adigun, 2004; Dilyana and Petya, 2002; Hagimont,and Ismail, 1999; Mawhood-Yunis, Nayak, Nussbaum, and Santoro 2004; Oke, Okelana, and Aderounmu, 2002; Olajubu, Aderounmu, and Akinde, 2004; and Oluwatope, Aderounmu, Akande, and Adigun ,2004.

In this work however, we attempt to compare the mobile agent and client server paradigms both in java environment using RMI for client server. The performance measurements simulated are the invocation cost, search time and fault tolerance.

RMI Communication Model

RMI is the object equivalent of RPC. RPC calls procedure over a network whereas RMI invokes object methods over a network. The server defines objects that client can use remotely and clients can now invoke methods of remote object as if it were a local object running in the same virtual machine as client. RMI possesses the following characteristics

- hides the underlying mechanism of transporting method arguments and return values across network,
- uses a network based registry program to keep track of the distributed object.
- a simple name repository (registry) that acts as a central management point, and
- server makes methods available for remote invocation by binding it to a name in the registry.

The underlying mechanism actually follows the following steps:

- The client invokes a local method called the client stub
- The client stub parcels up the client parameters to the remote procedure, converts them to a standard format and builds network messages (**marshaling)**. RMI uses object serialization to marshal and de-marshal messages.
- The network messages are sent to the remote server by the client stub through a previously agreed upon protocol.
- On the remote server, the server skeleton de-marshals the parameters from the network messages and converts them.
- The skeleton invokes local methods to invoke the actual server function, passes the parameters it received on the network messages from the client stub.
- When the server procedure completes, it returns to the skeleton with return values.
- The skeleton converts the return values, if necessary and marshal them into network messages to send back to the client
- The messages are transferred across the network to the client stub
- The client stub de-marshals the return values to the client.

Mobile Agent Paradigm

Mobile agent technology facilitated by recent advances in computers, communications and artificial intelligence, provides an attractive framework for the design and implementation of communicating applications in general and distributed knowledge networks in particular. The computational model based on mobility, can be seen as a replacement, refinement or extension of the traditional client server paradigm. The mobile agent framework emerged in the pursuit of open and decentralized models relevant to the dynamic and distributed nature of computations on the Internet.

Mobile agent is an object that migrates through many nodes of a heterogeneous network of computers under its own control in order to perform tasks using resources of these nodes (Aderounmu, 2004). The user initiates the agent, provides the agent with an itinerary (route), which represents the nodes to visit. The agents take the itinerary data, execution state and necessary code as part of a “bag” for starting its journey on each server. Once the mobile agent finishes its activities on the server, it determines dynamically the next host to visit on the network. As soon as this is done, the mobile agent sets up a client connection routine, which is a subcomponent of the mobility infrastructure in order to establish a link between the current node and its destination node. The distributed mobility infrastructure on each machine serves as the server for the client to establish communication channel that is based on Transmission Control Protocol/internet Protocol (TCP/IP) sockets. After establishing the connection, mobile agent moves through the mobility mat to the destination machine where it is enabled again to perform its activities.

The mobile agent queries the local environment to acquire the necessary information to achieve its objectives. The information retrieval service (corresponding to the current node) interacts with its own database management systems and returns the results to the agent, which adds them to its “bag” and migrates to the next host.

In case a server is disconnected or faulty, the agent skips the node, determine alternative route dynamically and continues its journey. It takes note of any problems and accomplished activities. When all nodes have been visited the agent sends a message to the original host with all results obtained and reports of its itinerary.

Analysis of Performance Parameters

Simulation program was developed to provide performance analysis between the RMI and MA based on communication models explained in section 2. The following parameters were used: search time, fault tolerance, and invocation cost.

Search Time

The time interval between when a query is sent from the source and when the response (data) is found and received at the requesting host is referred to as the search time in this research work. The search time is measured in terms of request time**,** $\mathbf{t_{req}}$ and response time, $\mathbf{t_{res}}$, relative to the distance between the nodes. In this paper, the sizes of request and response are assumed to be the same and thus request time and response time are the same and correspond to $\mathbf{t_r}$.

With RMI, when a query is generated, client connects to a remote object that was previously registered on the server, searches through the registered object on the server and reports back to the client. If the required information or object is not found on a host, a report is sent to the client before it connects to another server to search for the same object. It reports back to the server again before connecting to yet another server until the object is located or otherwise. If we assume the following:

- the distance between all the nodes is equal.
- the time taken to send a query is t_{req} , and
- the time the response is received at the requesting host is t_{res}

then the search time for one request is the time required to transmit the request from the source node to the destination node and back to the source. This is given by:

$$\mathbf{ST_{rmi}} = t_{req} + t_{res} \qquad 3.1$$

Where

1. $\mathbf{ST_{rmi}}$ is the search time for RMI
2. $\mathbf{t_{req}}$ is the request time
3. $\mathbf{t_{res}}$ is the response time

Each time the RMI visits a node it reports back to the requesting host (client), assuming there are **n** number of nodes in the network, and **m** denotes the number of nodes visited during the search, including the launching (client) node, i.e. the object is found at the **m**^th^ mode, where $m \leq n$, then

$\mathbf{ST_{rmi}} = m\,(t_{req} + t_{res})$

Assuming size of request is equal to the size of response and equal to t_r

$\mathbf{ST_{rmi}} = 2*t_r(m-1)$ ……………………………………………………….(3.2)

Where

- $\mathbf{t_r}$ is the request/response time
- n is the total number of nodes connected on the network, and
- m is the number of nodes visited to locate the object

With mobile agent on the other hand, the client creates an agent which contains the client's request to be executed: this agent moves to server A for local interaction, if the object is not found, the agent moves to the next server, B and yet another server until the object is found. Otherwise, until all the servers are visited, before reporting to the client.

The agent visits each node in $\mathbf{t_r}$ time, using the same notations, as with RMI, and let us

assume the size of agent code is **y.** If it takes **ty** units of time to transfer the agent code on the network. The search time of mobile agent paradigm for one request will be equal to the time to transport the request and agent code on the network between the nodes, thus,

$\mathbf{ST_{ma} = t_r + ty}$ **--- (3.3)**

Assuming the resource is found at node **m,** which includes the launching node where $\mathbf{m \le n}$**,** the total search time thus becomes

$\mathbf{ST_{ma} = (m-1)(t_r + ty)}$ **--- (3.4)**

At the end of the search the agent returns to the source with the response in another t_r + ty time. Such that the total time becomes

$\mathbf{ST_{ma}} = m(\mathbf{t_r} + \mathbf{ty})$ ……………………………………………………………………(**3.5**)

Invocation Cost

Cost is estimated in terms of the number and type of invocations involved. The invocations involved in both schemes are classified as either network (remote) invocation or local invocation; we thus generate a cost function for the two schemes.

The RMI scheme involves several invocations between the client and the server on the network. In RMI, once a client object connects to a remote object that was registered on a server, the connection has to be maintained to maintain execution. RMI makes remote invocations as the name implies, every invocation is over the network. The request is sent to the server over the network and the response is sent back to the client the same way. The request is taken one after another and sent to the server with the server sending the response each time across the network. If there are n requests (query) there will be n invocations to the destination and n invocations back to the source.

If we assume a request of size X bytes, response of size Y bytes and the cost of one request over the network (network invocation) is Ni, then, for n number of requests, each has a network invocation to its destination and one response over the network back to its origin, hence the total costs will be

$C_{rmi} = nX + nY$

if we assume that the request and response are the same size for the purpose of simplicity, i.e.

X = Y, then the total cost becomes:

$\mathbf{C_{rmi}} = 2nX$

and taking the request and response size to be equivalent to the cost of Network invocation Ni, i.e.

X = Ni

$\mathbf{C_{rmi} = 2nNi}$ ……………………………………………………………………(3.6)

The mobile agent paradigm allows remote invocations to be replaced by local invocations as the agent migrates to the invoked server site. The agent migrates with the execution code and execution thread. The execution thread enables the agent to trace its execution state and execute accordingly, i.e. continues from the last execution state instead of starting execution all over again on another host. In mobile agent scheme, one network invocation would carry all the requests and mobile agent's code and then execute at the remote server. All other invocations are mere local invocations on the server system. The mobile agent system makes fewer remote invocations compared to RMI with a lot of local invocations.

For n number of requests with size X bytes, the cost of network invocation will be nX.

The response to the requesting host is also of size Y bytes

If the cost of one local invocation is Li units, assume each request has one local invocation, n requests will be

nLi units local invocations at the invoked server site.

Since the agent moves with its code and execution thread, the size of the mobile agent code will have to be included.

Let the size of the mobile agent code be C

Then the total cost for the mobile agent system will be the sum of the total cost for network invocations, the total local invocations on the invoked server, the mobile code and the response cost over the network.

$$\mathbf{C_{ma}} = nX + nLi + C + Y \text{ -- } (3.7)$$

If the assumption that the size of request and response over the network are equal and equivalent to the cost of network invocations still holds, then, the total cost become:

$$\mathbf{C_{ma}} = (n+1)X + C + nLi$$

This gives

$$\mathbf{C_{ma}} = (n+1)Ni + C + nLi \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.8)$$

Where n is the number of requests

X: the size of request over the network

Y: the size of response over the network

Li: the cost of local invocation

C: the size of t he mobile agent code.

Fault Tolerance

Fault tolerance is the ability of a system to respond gracefully to an unexpected hardware or software failure. A fault tolerant system, as in computer networks, has the ability to continue operation in the event of a network failure.

Fault tolerance is a measure of robustness or adaptability of a system to breakdown. A fault tolerant system degrades gracefully in the face of failure, even though at a lower level of performance (Aderounmu, 2001). When there is a failure of a node in the RMI system or network connection failure, the RMI fails immediately. This is because most of the transactions involved take place over the network. The RMI will have to wait until the link is up again to continue execution.

In mobile agent system however, the agent proactively determines alternative route to the next host or source when failure occurs. The agent dynamically determines alternative path on getting to the host computer. The path is refreshed when moving to a new destination.

In carrying out our performance measure of fault tolerance for MA and RMI schemes, we assume the probability of failure to be 0.1, that is, in every 10 nodes there is likelihood of fault occurring in one out of 10. As in search time analyzed above, if we assume a network with **n** number of nodes, the size of request / response is assumed to be the same and thus the request time and response time are the same and equals $\mathbf{t_r}$**.** If we take the size of agent code to be **y,** then we assume it would take **ty** units of time to transfer the agent code on the network.

If we further assume that the recovery time for a failure is the same irrespective of the node at which the error occur and is given by $\mathbf{t_f}$, then probability that there is a failure at node **i** on the network is:

Pr (failure at i for i = 1-n) = 0.1

The node at which the fault occurs is generated as a random event that the system does not known a priori.

In MAS, the agent has a predetermined movement pattern at the starting node. The routing table is generated as a random process and it consist of the sequence of nodes on the network that the agent would traverse during its itinerary. However, in the event of a network or node failure, which makes the agent not to follow the original path, an alternative path is generated so as to bypass the failing node. This in effect helps to eliminate the downtime that the system should have experienced while waiting for the link to be up again. This is illustrated below:

Assuming that the mobile agent is moving on a network, with nodes labeled, 1, 2, 3, - ---, 10, and the routing pattern is given as:

$2 \rightarrow 5 \rightarrow 1 \rightarrow 4 \rightarrow 8 \rightarrow 3 \rightarrow 7 \rightarrow 9 \rightarrow 10 \rightarrow 6 \rightarrow 2$

Assuming fault occur between node 4 to 8, then an alternative path is generated at node 4 which may look like this
$4 \rightarrow 7 \rightarrow 3 \rightarrow 10 \rightarrow 6 \rightarrow 8 \rightarrow 2$.

If the resource being searched for is found at node number **m** (where $\mathbf{m \leq n}$) then the total search time ($\mathbf{T_{ma}}$) for MA for the resource is the time it takes to transport the requests and agent code on the network between the nodes, which is:

$T_{ma} = m\,(t_r + ty)$ ……………………………………………………………(3.9)

Where

m is the number of the node at which resource is found,

$\mathbf{t_r}$ the request/response time, and

ty the time taken to transfer the agent code

For RMI, there is no way the system can adjust dynamically to the failure. Hence, the system suffers a delay equivalent to the time at which the fault is rectified. Thus, the total search time ($\mathbf{T_{rmi}}$**)** is equivalent to the time it takes to transmit the request from the source node to the destination node and back to the source. This process is repeated until node **m**.

However, if a faulty node is encountered during the search, the system experience a delay equals to the total failure recovery time $\mathbf{kt_f}$ **.** Where **k** is the no of nodes at which error occur along the itinerary.

Thus:

$\mathbf{T_{rmi} = 2 * t_r\,(m - 1) + kt_f}$ **…………………………………………………………….(3.10)**

Where

k is the number of node at which failure occurs

$\mathbf{t_f}$ is the down time or the total failure recovery time

System Simulation and Result Analysis

Cost versus Number of Requests per Service

In this paper, we measured the invocation cost for varying number for requests per service as discussed in the last section. The result from our simulation also confirms that Mobile Agent approach results in reduced cost overhead than what are obtainable with the RMI scheme.

As shown in Figure1, the cost rises as the number of requests increases, and the cost of RMI increases with a higher gradient compared to mobile agent. The mobile agent paradigm incurs a lower invocation cost compared to RMI at the same or even better efficiency, a desirable quality of any system.

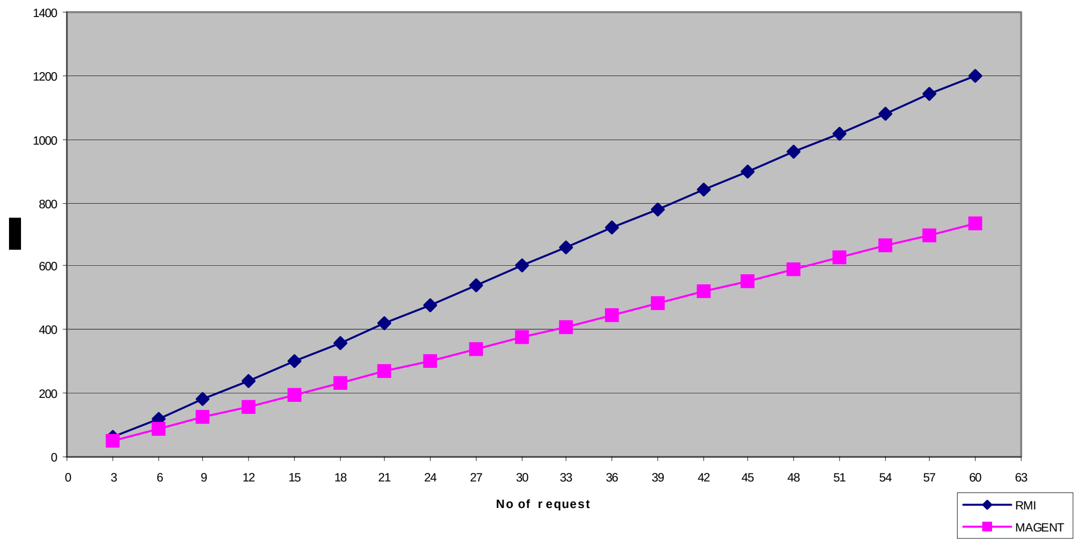


Figure 1 Graph of Cost versus number of requests

Search Time versus Number of Nodes

In measuring the search time involved in using the two approaches, we have compared the time it takes to search and locate a resource on the network. The location of the resource is not known a priori. Using the Mobile Agent approach, it took a shorter time to search for the resource than we have in RMI. Fig.2 shows that mobile agent paradigm offers a lower search time compared to RMI, despite the fact that it travels with the execution code. It should be noted that the search time does not depend on the number of nodes in the network, but the location of the resource being sought.

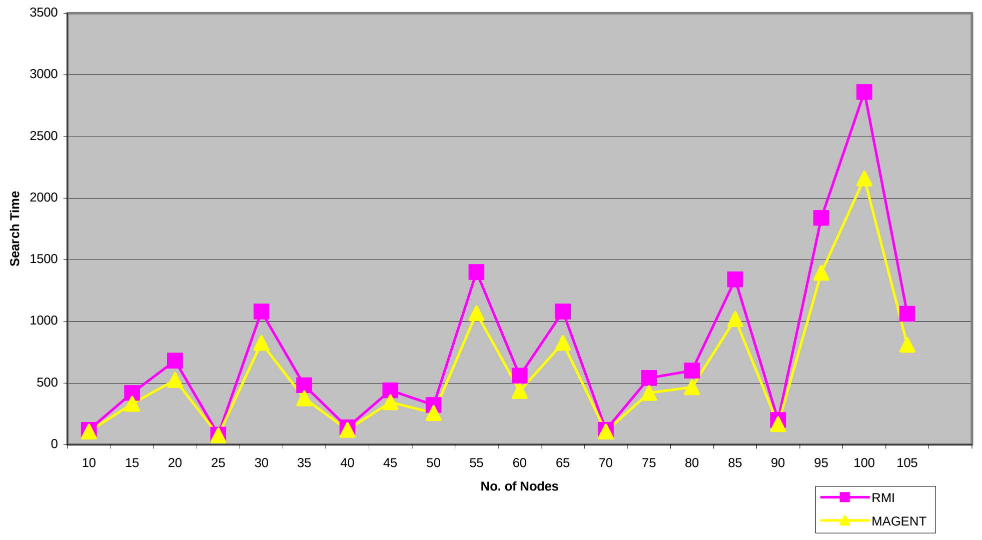


Fig.2 Graph of Search Time Versus Number of Nodes

Query Time versus Number of Nodes

In measuring the response of the MAS and RMI to fault, we measured the time it takes to complete a query for a given number of nodes. Our result indicates that as we have more nodes on the network, the more the likelihood of fault occurring and thus resulting in higher query time as is been experienced in RMI. However, MA is able to adjust in case of a failure and therefore reduce query time because it could dynamically generate alternative path for its itinerary.

Figure 3 shows that the search time for RMI increases in the face of node or network failure due to the down time. The surpassing performance of mobile agent paradigm can be connected to more local invocations it makes as against the RMI which relies heavily on network connections throughout the service, therefore making it more vulnerable to network failure.

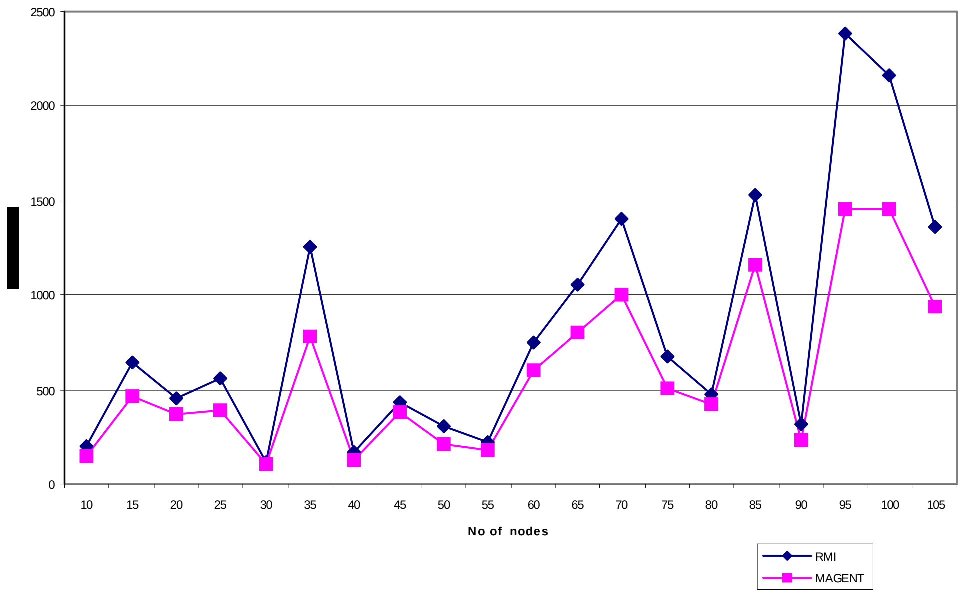


Fig.3 Graph of Query Time for different number of nodes

Conclusion

The authors present a comparison of the performance of MA with RMI for distributed information storage and retrieval. Our conclusion was based on the results of simulation carried out in this research work. The evaluation of the mobile agent paradigm with the client server reveals that the mobile agent paradigm is superior in terms of invocation cost reduction, reduction of search time and better adaptability to failures by dynamically defining alternative route in the face of network failure. However, our analysis does not account for searching large database

We are currently working on full implementation of mobile agent for information storage and retrieval. The perspective of future work in this regard is in the provisioning of adequate security for the mobile agent.

References

Aderounmun, G.A. (2004). Performance comparison of remote procedure calling and mobile agent approach to control and data transfer in distributed computing environment. *Journal of Network and Computer Applications*, 113 – 129.

Aderounmu, G. A., Adagunodo, E. R., Ogwu, F. J., & Akinde, A. D. (2000). On the concept of mobile agent framework for resource management in an enterprise network. *Proceedings of the IASTED International Conference on Artificial Intelligent and Soft Computing.* 168 – 172, July 24-26, Canada.

Aderounmu, G.A, Adekiigbe, A., & Iyilade, J.S.(2004). An evaluation of mobile agent paradigm. *The Journal of Computer Science and Its Applications, 10*(2): 107-116.

Ajayi, A. O., Aderounmu, G. A., & Adigun, M. O. (2004): Web-based expert system for property valuation using mobile agent technology. *Proceedings of the IASTED International Conference on Knowledge Sharing and Collaborative Engineering*, November, 26-31, St. Thomas, Virgin Islands. USA

Brewington, B., Gray, R., Moizumi. K., Kotz, D., Cybenko, C., & Rus, D. (1999). Mobile agents for distributed information retrieval. In M. Klusch (Ed.), *Intelligent information agents* (Chapter 15). Springer-Verlag. 355- 395. Retrieved 29/11/2005 from http://agent.cs.dartmouth.edu/papers/smallbib/brewington:IR.bib

Dale, J. & DeRoure, D. (1997). A mobile agent architecture for distributed information management. *Proceedings of the International Workshop on the Virtual Multicomputer*. Retrieved from http://www.mmrg.ecs.soton.ac.uk/publications/archive/dale1997b/

Dilyana, S. & Petya G., (2002). Building distributed applications with Java mobile agents. *International workshop NGNT* (pp 103- 109). Retreived from http://saturn.acad.bg/bis/pdfs/16_doklad.pdf

Hagimont, D.& Ismail, L. (1999). A performance evaluation of the mobile agent paradigm. *ACM SIGNAL Notices, 34*, 306--313.

Ismail, L. & Hagimont, D. (1999). A performance evaluation of the mobile agent. *Proceedings of the 14th ACM SIGPLAN conference on OOP Systems Languages and Applications.* Denver, Colorado, US. 306- 313.

Kunal, M. S. (2003). Performance analysis of mobile agent in wireless internet application using simulation. (M.Sc Thesis, Faculty of the college of Graduate Studies Lamar University).

Lange, D. B. & Oshima, M. (1999). Seven good reasons for mobile agents. *Communications of the Association for Computing Machinery, 42*(3): 88 - 92

Mawhood-Yunis, A., Nayak, A., Nussbaum, D, & Santoro, N. (2004). Comparing performance of two mobile agent platforms in distributed search. *IEEE/WIC/ACM International Conference On Intelligent Agent Technology* (IAT' 04), pp 425 – 428.

Oke, A. O, Okelana, O. M, & Aderounmu, G. A. (2002). A mobile agent system for electronic shopping. In *Proceedings of the ICT applied to Economic Intelligence* (ICTEI), 105 – 114. August 20-23, Nigeria.

Olajubu, E.A., Aderounmu, G.A., & Akinde, A.D (2004). An agent-based architecture for students' results messanger, Obafemi Awolowo University as case Study. *The Journal of Computer Science and Its Applications, 10*(2): 3-16

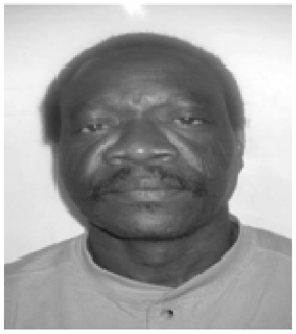
Oluwatope, A. O., Aderounmu, G. A., Akande, O. & Adigun, M. O. (2004). An auction scenario in a stock market using mobile agent. *Southern Africa Telecommunication Networks and Applications Conference*, 229 – 234. Western Cape, South Africa.

Seng, W. L. (1999). Mobile agent technology for enterprise distributed applications: An overview and an architectural perspective. *CRC for Distributed System Technology*, Monash University, Caulfield Campus, Caulfield East, Victoria 3145, Australia.

Biographies

G.A. Aderounmu holds a research degree M.Sc./PhD in Computer Science from Obafemi Awolowo University, Ile-Ife, Nigeria (1997 and 2001, respectively). He is a member of the Nigerian Society of Engineers (NSE) and is also a registered computer engineer with Council for the Regulation of Engineering Practice in Nigeria (COREN). He is also a Member of Nigerian Computer Society (NCS) and Computer Professional Registration Council of Nigeria (CPN). He has over 13 years of experience in teaching and research. He is an author of many journal articles in Nigeria and abroad. His special interests include engineering education in Nigeria, curriculum development, and computer communication and network. He is a Visiting Research Fellow to the University of Zululand, Republic of South Africa. He is currently a Senior Lecturer, Deputy Director of the Information Technology and Communication Unit (INTECU) and the Acting Head of the Department of Computer Science and Engineering of the same university.

B. O Oyatokun obtained her research degree M.Sc. Computer Science from University of Ibadan, Ibadan, Nigeria in 2004. Her current research interests are in the areas of computer communications and deployment of mobile agent software management of heterogeneous computer network. She is currently a Ph.D student at the Department of Computer Science , University of Ibadan.



M. O. Adigun holds a research degree PhD in Computer Science from Obafemi Awolowo University, Ile-Ife, Nigeria which he obtained in 1989. Currently, he is a Professor and Head, Department of Computer Science, University of Zululand, Republic of South Africa. He is an author of many journal articles in Nigeria, Republic of South Africa and abroad. His research interests include Software Engineering, Mobile Computing, Modeling and Simulation, and Performance Analysis of Computer System.