Journal of Computing and Information Technology - CIT 19, 2011, 2, 93–104
doi:10.2498/cit.1001375

Lightweight Agents, Intelligent Mobile Agent and RPC Schemes: A Comparative Analysis

Emmanuel A. Olajubu, Ganiyu A. Aderounmu and Anuoluwapo O. Ajayi

Department of Computer Science and Engineering, Obafemi Awolowo University, Nigeria

This paper presents the performance comparison of Lightweight Agents, Single Mobile Intelligent Agents and Remote Procedure Call which are tools for implementing communication in a distributed computing environment. Routing algorithms for each scheme is modeled based on TSP. The performance comparison among the three schemes is based on bandwidth overhead with retransmission, system throughput and system latency. The mathematical model for each performance metric is presented, from which mathematical model is derived for each scheme for comparison. The simulation results show that the LWAs has better performance than the other two schemes in terms of small bandwidth retransmission overhead, high system throughput and low system latency. The Bernoulli random variable is used to model the failure rate of the simulated network which is assumed to have probability of success p = 85% and the probability of failure q = 15%. The network availability is realized by multiplicative pseudorandom number generator during the simulation. The results of simulation are presented.

Keywords: throughput, retransmission overhead, latency, Bernoulli random variable, distributed computing

1. Introduction

The evolution of computer network has brought a great improvement to data communication and it is now a veritable means of data transmission [1-2]. The modern data communication network is increasingly becoming more complex, and in this information age, impressive growth is expected in the use of communication networks, not only in the developed world but also in the developing countries, therefore more intelligent application software will be required for effective and efficient data transfer in the distributed computing environment. The present data communication networks are usually conglomerates of many heterogeneous, very often incompatible, multi-vendor components which call for intelligent applications for moving data across different networks. The enormous increases in the size and complexity of these communication networks are fueling the increase in research for intelligent network application systems. The main objective of this intelligent distributed network applications system is to improve efficiency, reliability and availability of the networks and provide a higher degree of performance in an efficient and cost effective manner. In this paper, performance evaluation of Remote Procedure Call (RPC), Single intelligent Mobile Agent (SMA) and Lightweight Agents (LWA) was carried out; it reveals some salient performance benefits of LWAs over both RPC and SMA for data transfer across distributed computing environment. We have considered these three schemes carefully, because they appeared as the main technologies used for implementing distributed systems. RPC is used as client/server paradigm while SMA is an intelligent self-contained and autonomous agent also used to implement distributed systems, and LWAs are tiny multi-agents that have limited intelligence in their problem solving domain. Communication infrastructure provided by multi-agents systems allows their interactions with each other which results in their global intelligent behavior. The agents in this system are entities with the ability to sense their immediate environment and undertake simple processing of environmental observations in or-

der to perform simple computational task. Their comparison highlights the advantages and disadvantages of using any of the schemes in a distributed environment.

The rest of this paper is organized in the following sections: Section 2 presents the concept of mobile codes. In Section 3, the TSP routing model for the agents is discussed. Section 4 discusses model formulation for Bernoulli Random Variable (BRV), Retransmission Overhead due to denial of service, system throughput and latency, while Section 5 is the mathematical models for each scheme considered in this work. Section 6 discusses system simulation model and experiment, Section 7 presents simulation result analysis for each scheme and finally Section 8 concludes the paper.

2. Communication Model

Code mobility in distributed computing environment subtly started with process migration which enables a process to transfer its execution between two or more machines or two or more processors in the same machine. Migrating processes in distributed computing environment is fully functionally dependent on the operating system. The advantages attributed to process migration includes: efficient load distribution which is achieved by moving processes from congested nodes to less busy nodes, increases fault tolerance of the system by moving processes away from nodes that may be experiencing partial failure, improve system administration by moving processes away from the node that is about to be off and, lastly, data are made locally available when processes are migrated to the source or closer to the source of data [7]. Despite the laudable advantages of process migration, [5] itemized some salient reasons why process migration is yet to gain widespread acceptance.

Distributed computing which is believed to have evolved from traditional dumb terminals hooked to the mainframe or central computer saddled with the sole responsibility of file processing. The improved processing capacity of personal computers (PC) allowed the possibility of sharing some processing resources between the mainframe and the PCs attached to it. This gave birth to the client/server architecture which offers better computing control [8]. The communicating entities have a well-defined role; the client sends request to the server, the server processes the request and informs the client when the process is completed. The result of the process is sent back to the client. This model makes client dependent on the server. One of the widely accepted applications of client/server paradigms is the RPC.

2.1. RPC communication model

RPC has gained some degree of acceptance since majority of Internet applications are based on it. Unlike classical client/server model that operates through message passing, which places the burden to determine network addresses and system synchronization points on the programmer, the client merely requests a service from the server and then waits for reply. The call message contains the parameters of the procedure and the reply message contains the procedure results.

2.1.1. Mobile agent concept

Mobile agent is defined as a computer software that possesses both code and data which is able to migrate (move) from one computer system to another on a heterogeneous communication network autonomously and continue its operation on the destination system. It assumes the combination of distributed computing technology and software agent technology. Here, flexibility is the core of the system. Any host on the network could possess the combination of know-how, resources and the processing ability. This system gives a good situation of open system and peer-to-peer communication procedure. Either of the communicating entities could be server or client at any situation, depending on the entities where resources are located.

2.1.2. Lightweight agents

LWAs are miniature mobile agents that have small intelligence in their domain, but their communication with each other and interaction with their environment is capable of exhibiting global intelligent behavior. The authors of [9]

opined that these agents are entities with the ability to sense their immediate environment and embark on computational task in the environment. A global intelligence comes out as these agents communicate locally among themselves. Their interaction with their immediate environment produces a global intelligence as end [3]. Lightweight agents can be described as agents that easily accomplish their goal with a very negligible code size. They are updatable and up-gradable at the site of operation, smaller, simpler, and faster to transport on communication network due to their minor code size. In distributed computing system scenario, there are different types of operating systems. For an agent to effectively perform data transfer in these systems, it needs to carry the necessary protocol for data retrieval and filtering which applies to the different systems. If the agent is designed not using LWAs concept, it will always carry all the protocols that are required for different systems and, therefore, the code size will be very large and it will be prone to network waste resources [4].

3. Routing Model

The cost effective path planning for the three schemes closely mimics the TSP which could be formulated as follows: an agent or a process wishes to visit N distinct nodes on the network and return to the home or source node. The latency between nodes k and k+1 on the network is given as $t^l_{k,k+1}$ and also, $t^l_{k,k+1} = t^l_{k+1,k}$. The agent or process assignment is to find the sequence of tours so that the overall distance traveled or cost of travelling is minimized. This is synonymous to mobile agent routing system of [9], but different from routing technique developed in [12]. Our model calculates the cost between two nodes on computer networks during simulation time rather than using routing table used in [13] which is prone to memory consumption and which may consequently slowdown the system. Figure 1 gives a simple hypothetical case for this model. The PC_1 is the source node for the scheme, and there are sub-tours within our hypothetical network example. When the agents start their itinerancy on node PC1, there are three alternative routes available to the agents. The routes are PC1 to PC2, PC1 to PC3 and, lastly, PC1 to PC4. The agents will select the shortest route which is set to 1 and every other route to the node is set to 0 by the artificial variable X, as shown in equation 1. Our agents used look ahead algorithms [14] to select the shortest path among various alternatives on communication network. It is worth noting that the agents can move in any direction (bidirectional) on the network so as to achieve their objectives. The main goal is to minimize as much as possible the amount of bandwidth usage and other associated costs. On the next system, i.e. PC2, there are also three alternative routes out of which the agents are to select the shortest path. The exercise is repeated as in PC1. Every node visited on the network in one itinerancy is assigned a digital signature which makes it impossible for another agent of the same group to visit the same node again. For simplicity and clear understanding, the digital signature is assigned B, while all unvisited nodes are assigned W. It means those nodes need to be visited. The purpose of digital signature is to avoid multiple visits to a node during one itinerancy which may increase bandwidth usage. The digital signature gradually disappears from the components before another round of system agents' itinerancy. Generally, the order in which the agents visit the nodes is of the form $i_1, i_2, i_3 \ldots i_N$ where N is the last node on the network. The direction of movement of the agents is not important since latency between node k and k+1 is given as $t^l_{k,k+1}$ and $t^l_{k,k+1} = t^l_{k+1,k}$.

Equation 1 has an artificial parameter which decides whether the path $t^l_{k,k+1}$ is feasible or not. If the path $t^l_{k,k+1}$ is 0, then it is not feasible, while if it is 1, the sub tour is feasible. Equation 2 is the maximum or the longest route that agent can take to accomplish its task on the network, while equation 3 describes the TPS routing pattern for the three schemes, which minimizes the total routing cost. The equation illustrates the typical TSP algorithm:

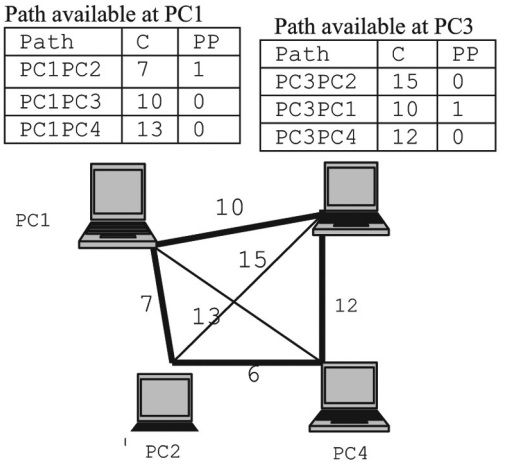
$$t^l_{k,k+1} = \begin{cases} 1 \text{ if the edge } t^l_{k,k+1} \rightarrow t^l_{k+1,k} \\ \quad \text{is in the tour} \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

$$\Phi_{\max} = Max(T(n_i)), 0 \leq i \leq N \tag{2}$$

$$Min.Z = \sum_{k=1}^{N} (t^{l}_{k,k+1} + t^{c}_{k}) X_{k,k+1} \leq \Phi_{\max} \quad (3)$$

$t^{l}_{k,k+1}$	Latency between node k and node k+1
t^{c}_{k}	Computational time on the node k
n_i	Total number of nodes on the network
T	Tour

Table 1. Notations summary for equations 2 and 3.



	PC1	PC2	PC3	PC4
Agent at PC1	X	W	W	W
Agent at PC2	B	X	W	W
Agent at PC3	B	B	X	W
Agent at PC4	B	B	B	X
	B	B	B	B

Figure 1. Simple routing scenario.

The last row indicates that all nodes on the network have been visited by the agents.

4. Model Formulations

4.1. Bernoulli random model

A Bernoulli trial is an experiment in which only two outcomes are possible. S={success, failure} success, with probability ρ, and failure, with probability $1-\rho$. If X is a discrete random variable, which is stochastically based on the BRV, X will take values of 1 or 0, which is network availability and network outage respectively.

Therefore, X is considered the random variable defined by

$$\mathrm{X\ (failure)} = 0$$
$$\mathrm{X\ (success)} = 1$$

Suppose that P (success)=p.

The probability mass function of X is then given as:

$$\rho(0) = 1 - \rho \quad (4)$$

$$\rho(1) = \rho \quad (5)$$

Or rather

$$\rho(x) = \rho^{\mathrm{x}}(1-\rho)^{1-\mathrm{x}} \quad (6)$$

Where x = 0, 1

With the normal axiom for probabilities of a discrete random variable X with X= $\{x_1, x_2\}$:

$$\sum_{i=1}^{\infty} \rho_x(x_i) = 1 \quad (7)$$

The cumulative distribution function is given as:

$$F_x(a) = 1 - \sum_{x=a}^{\infty} \rho_x(x). \quad (8)$$

For the peculiar case of Bernoulli random variable:

$$\sum_{j=0}^{1} \rho_x(x_1) = 1 \quad (9)$$

Since j assumes value of 0 and 1, then we define the parameters p and q where: P= $\rho_x(1)$ ranges between $\{1, 0\}$; and q=p. When $\{x = x_1 = 1\}$ is the network availability with the probability of P, while the virtual network used in modeling this work has a state that is model after the BRV with P=0.85 and q=0.15. This shows that the network has a downtime of 15%. The network outage was simulated by multiplicative pseudorandom number generator with probability distribution that follows the BRV p=0.85 and q=0.15.

4.2. Retransmission overhead due to denial of service

Let $f_A^0(t)$ be a time variate function which represents denial of service at time t for the system;

And $f_A^1(t)$ is a time variate function for successful service at time t for the same system. Then, $\sum_{t=0}^{t^n} f_A^0(t)$ equals the total number of denials of service between simulation times t_0 and t_n for the system. Likewise, $\sum_{t=0}^{t^n} f_A^1(t)$ equals the total number of successful messages within the time between simulation times t_0 and t_n for the system.

Therefore, the total number of messages sent within the time is:

$$= \sum_{t=0}^{t^n} f_A^0(t) + \sum_{t=0}^{t^n} f_A^1(t)$$
$$= \sum_{t=0}^{t^n} (f_A^0(t) + f_A^1(t)) \qquad (10)$$

The percentage of message failure is therefore given as:

$$\frac{\sum_{t=0}^{t^n} f_A^0(t)}{\sum_{t=0}^{t^n} (f_A^0(t) + f_A^1(t))} \cdot 100 \qquad (11)$$

For the system between times span t_0 and t^n.

It should be noted that Figure 3 represents mobile agent, lightweight agents or RPC systems. It also assumes that f is directly related to the probability distribution of p and q.

4.3. System throughput

Software throughput is generally defined as the amount of data the software can process within a unit of time. Also, [6] defined throughput as the number of messages the system can handle within a sample interval of time. Therefore, the researcher defined software throughput in this system as number of detected faults that are funneled to the network engineer within a unit of time.

Let $f_m^o(m)$ be a time variate function representing the number of messages the system can send within an interval of time for this system and $f_m^1(m)$ is the time variate representing the number of messages unsuccessfully delivered.

$\int_{t_0}^{t_s} f_m^o(m)dt$ represents the total number of messages sent between simulation times t_0 and t_s for the system.

Likewise $\int_{t_0}^{t_s} f_m^1(m)dt$ is the total message unsuccessfully delivered between the simulation times t_0 and t_s for the system.

Therefore, the total message successfully delivered within the simulation time is given as:

$$M_s = \int_{t_0}^{t_s} f_m^o(m)dt - \int_{t_0}^{t_s} f_m^1(m)dt \qquad (12)$$

Where M_s is the number of messages successfully delivered within the time frame.

Hence, the system throughput is given as follows:

$$T_s = \frac{\int_{t_0}^{t_s} f_m^0(m)dt - \int_{t_0}^{t_s} f_m^1(m)dt}{t_s - t_0} \qquad (13)$$

4.4. System latency

System latency is defined as the period of time it takes a packet to travel from source to destination. In this work, system latency is defined as the time interval in between when the agent senses a fault and the time the fault is interpreted by FI and then funneled to the network engineer's desktop. This is the aggregate delay in receiving a fault message due to traffic, overhead, etc.

Fault message latency is given as:

$$l_f = T_t - t_{id} \qquad (14)$$

Where $t_{id} = \frac{\mu_l}{\nu_s}$ substituting this into the above equation we have

$$l_f = T_t - \frac{\mu_l}{\nu_s} \qquad (15)$$

Where:

l_f = fault message latency
T_t = total round trip time
t_{id} = ideal time for fault message transfer
μ_l = fault message length
v_s = Network speed.

5. Mathematical Model Derivation for each Scheme

5.1. Load across the network

SMA Load across the network

For SMA, the total load transferred across the network for visiting all the network elements is expressed in terms of its code size and the result of its query. During agent itinerancy to the node, the message size is zero (i.e. y=0) since the agent only migrates to the nodes with its code size X_{sma} and the agent returns with the same code size, therefore the code size for the complete itinerancy is $2X_{sma}$ and the number of nodes on the network is N. But the result of its search from the nodes is represented as Z bytes, then the total load across the network is given as:

$$\eta_{sma} = (2X_{sma} + Z)N \tag{16}$$

LWAs Load across the network

As in the case of SMA, the agents visit the nodes with their code size X_{lwa} and as it returns to the home node, the total code size becomes $2X_{lwa}$. The message size y=0 and the result of its search from the nodes is given as Z bytes. Therefore, the total load across the network is given as:

$$\eta_{lwa} = \sum_{i=1}^{k} (2X_{lwa} + z)_i N \tag{17}$$

RPC Load across the network

Let the message size from the client be denoted by y in bytes and the size of reply from the server be z in bytes, for N nodes the total load across the network is given as:

$$\eta_{rpc} = 2yN \tag{18}$$

5.2. Response time of the models

Response time for mobile agent is as shown in Eq. (19):

$$MA^t_{res} = \sum_{k=1}^{N} (t^l_{k-1,k} + t^c_k) \tag{19}$$

Response time for RPC is denoted as Eq. (20):

$$RPC^t_{res} = \sum_{k=1}^{N} (t^l_{k-1,k} + t^c_k) \tag{20}$$

Response time for Lightweight Agents is presented in Eq. (21):

$$LWA^t_{res} = \frac{\left(\sum_{K=1}^{N} (t^l_{k-1,k} + t^c_k) \right)}{A_i} \tag{21}$$

These equations were formulated in [9].

5.3. Mathematical model for retransmission overhead

For SMA scheme, if η^0_{sma} represents message failure and η^1_{sma} denotes message successfully delivered between time t_0 and t_n and ReT_{sma} denotes retransmission overhead, when these variables are substituted into equation Eq. (11), we have:

$$ReT_{sma} = \frac{\sum_{t=0}^{t^{tn}} \eta^0_{sma}(t)}{\sum_{t=0}^{t_n} (\eta^0_{sma}(t) + \eta^1_{sma}(t))} \tag{22}$$

Similarly, LWAs' retransmission is also denoted as ReT_{lwa} with the same variables as in the case of SMA:

$$ReT_{lwa} = \frac{\sum_{t=0}^{t_n} \eta^0_{lwa}(t)}{\sum_{t=0}^{t_n} (\eta^0_{lwa}(t) + \eta^1_{lwa}(t))} \tag{23}$$

And, finally, RPC model for retransmission is derived from Eq. (11) in the same way for SMA

and LWAs respectively, as shown in Eq. (24).

$$ReT_{rpc} = \frac{\sum_{t=0}^{t_n} \eta^0_{rpc}(t)}{\sum_{t=0}^{t_n} (\eta^0_{rpc}(t) + \eta^1_{rpc}(t))} \quad (24)$$

5.4. System throughput

The system throughput can be defined as the quantum of data the software can process within a unit of time. During data transfer across the network, in practical their band to be network failure which will result in denial of service. This denial of service will definitely affect the system throughput i.e. the number of messages that can be processed within a unit of time. The system throughput mathematical model for SMA was derived by substituting Eq. (16) into equation Eq. (13) which gives:

$$T_{s(sma)} = \frac{\eta_{sma}}{t_s - t_0} \quad (25)$$

Also, the system throughput for LWAs' model was derived by substituting Eq. (17) into equation Eq. (13) which gives:

$$T_{s(lwa)} = \frac{\eta_{lwa}}{t_s - t_0} \quad (26)$$

Lastly, the RPC mathematical model was developed by substituting Eq. (18) into equation Eq. (13) which gives:

$$T_{s(rpc)} = \frac{\eta_{rpc}}{t_s - t_0} \quad (27)$$

5.5. System latency

The system latency is to determine the inherent processing speed of the schemes. This will have impact on the number of information the system can process in a unit of time. Consequently, it will affect the response of the system. Response time is very crucial in any information search and retrieval environment. The system latency is simulated for the three schemes against the number of nodes on the network.

To build simulation model for SMA response time, equation 16, which is the average load of the scheme and equation 19 which is the observable latency of the system (including computer network latency) are substituted into Eq. (15), the real system latency. The equation becomes:

$$l^{sma}_f = MA^t_{res} - \frac{\beta_{sma}}{100kb/s} \quad (28)$$

Likewise, the simulation model for LWAs is carried out by substituting Eq. (17) and Eq. (21) into Eq. (15) this gives:

$$l^{lwa}_f = lwa^t_{res} - \frac{\beta_{lwa}}{100kb/s} \quad (29)$$

And, finally, the RPC simulation model is done by substituting Eq. (18) and Eq. (20) into equation (15). The equation becomes:

$$l^{rpc}_f = RPC^t_{res} - \frac{\beta_{rpc}}{100kb/s} \quad (30)$$

We assume 100kb/s for the speed LAN used in the simulation.

6. System Simulation Model

In this section, a simulator Figure 2 is presented to simulate the communication model of the three schemes discussed in Section 2. This simulator consists of five distinct modules which are: System Objects consisting of LWAs systems which is controlled by Lightweight Agents Controller (LWAC), SMA and RPC systems; BRV: Network failure generator; computer network size; processing unit and output unit. In Figure 2, the computer network size is the computer network where the three schemes are expected to carry out information search and retrieval on each node. The network size is expected to be dynamic due to mobile devices that enter and leave the network at random. The BVR: network failure generator is an object that indicates network availability and failure by generating signals that represent network failure to the simulation architecture. The system object is the component that consists of the three systems to be simulated. The processing unit processes the traffic information while the output unit gives the result of simulation in human language.

The mode of operation of this simulation architecture shows that there is network failure.

The RPC sends request to different nodes where information search and retrieval are required and SMA and LWAs migrate to the network to visit individual node where computation will be done. The communication model described in Section 2 would have continued uninterruptedly, but for network failure. When there is network failure, retransmission will be necessary to complete the whole process so that the service is successively completed.

During the simulation work, the following parameters were defined [5].

Process. A process is the standard representation for different activities carried out to successfully complete a task. In information storage and retrieval scenario, it may involve searching, filtering and sorting information from a database.

Request. A process is usually made up of a series of request and reply cycles. In RPC, the client forwards request sequentially and reply is also sent back sequentially. Normally, more than one request are necessary to complete a service.

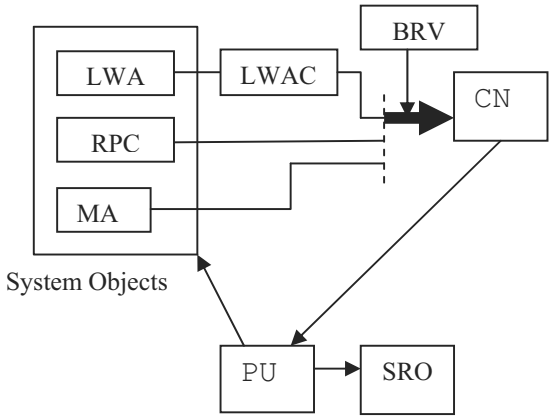
Network Failure Rate. This is the parameter that determines network reliability. This is measured as the percentage of non availability of network connections for a specified period of simulation time.

Retransmission. It is the process of resending a service that could not be concluded due to network connection failure.

Fault Tolerance. This shows the degree to which the system can accommodate network breakdown. A fault tolerant system will employ grateful degradation at the time of failure though the network performance may dwindle.

Message size. It is the size in bytes of a single request or reply.

Within the context of theoretical framework for this paper, we demonstrated the influence of network outage on network performance by using BRV to model network failure during operational time. In practical sense, there will be intermittent network failure in any network, either due to link failure or network traffic congestion; this may reduce the network performance and consequently affect management software



Legend

LWA	Lightweight Agents System
RPC	Remote Procedure Calling System
MA	Mobile Agent System
LWAC	Lightweight Agents Controller
BRV	Bernoulli Random Variable
CN	Computer Network
PU	Processing Unit
SRO	System Result Output

Figure 2. Simulation architecture.

on such network. It may increase retransmission overhead and reduce the system throughput. In our simulation, we assume 15% downtime for the simulated network. We also employ a pseudorandom number generator that has a time variate probability distribution of Bernoulli Random Variable (BRV) which is appropriate for simulating network of this status [5]. The BRV can only accept two inputs, one or zero, these two state input correspond to network availability or network failure. This observable fact that is stochastic in nature is easily expressed with probabilistic distribution using BRV.

Simulation Experiment

A dynamic virtual network whose network elements range between 100 nodes to 1000 nodes was set up in Java run time environment. The network allows the dynamism of the modern network as mobile devices can easily come in to the network and leave at any time. As discussed in Section 4.1, the simulation system was programmed so that BRV intermittently generate network failure during the simulation time. In our experiment, the RPC scheme employs RPC communication model to send requests to differ-

ent nodes on the network. During the simulation time network failure of 15% was assumed in our experiment. The model was allowed to communicate with all nodes on the network using TSP as routing protocol so that a node is not visited twice in an itinerancy and the cost in term of bandwidth (Kb/s) and time (s) were recorded. Also, SMA with all intelligence required was allowed to search and retrieve information in all nodes in the virtual network. LWAs were also deployed to the virtual network for information retrieval. In the case of LWAs, the network is partitioned to the number of agents in the group. The experiment was allowed to run, and three different readings were recorded for each performance metric. The values of the readings followed the same sequence but with little variation in the values. The average of the three readings was taken so as to minimize error. See Tables 1-3 as samples of the average readings, on this Table NTS is Network Size. This same experiment was carried out for each of the performance metric used to bench mark the schemes.

7. Analysis of Simulation Result

7.1. System bandwidth overhead due to retransmission

Retransmission is defined as the process of resending packets that could not get to the required destination(s) from the source node on the communication network. The mathematical model for percentage of message failure was developed in section three. From Eq. (10), as the denial of service increases, the need for retransmission becomes very obvious. This may be due to network failure or congestion resulting from too much traffic flooding the network. When this happened, there is the need to retransmit the dropped packets so that the service will be completed. This situation could unnecessarily increase the bandwidth overhead, if allowed unchecked in a system. In this research work, the retransmission due to network bandwidth usage is compared among the three schemes (i.e. RPC, Mobile Agent and Swarm Intelligence). The performance factor for this simulation is the percentage of the total message size that is retransmitted during an operation varying the number of nodes visited by each scheme. For simulation purpose in this work, the network failure rate is fixed at 15% and the message size from each node is assumed to be about 1 byte.

Consequent on the above analysis, Figure 3 was obtained through simulation varying the number of nodes **N** for each itinerancy. From this simulation, it is clearly shown that the retransmission in lightweight agents is far less than in the other two schemes. Although, batch message transfer of data across the network is used by both single mobile agent and the lightweight agent which promise a shorter-time data transfer across the network when compared with RPC, the distributive nature of lightweight agents gave it a better performance than the single mobile agent.

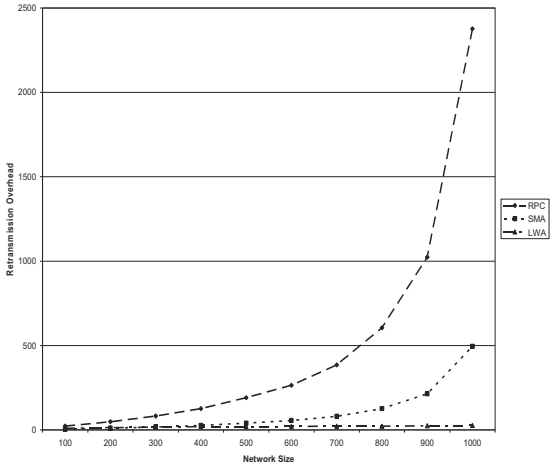


Figure 3. Retransmissions of the three schemes.

This is an expected result since the load is shared among the lightweight agents participating in information retrieval on the network. It is important to note that the higher the load, the longer the time it takes to push it across the network, making it more liable to network failure. In this simulation, as we vary the number of nodes on the network, the load to be pushed across the network increases for the three schemes.

7.2. System throughput

The objective of this experiment is to determine the general performance of the message throughput for each scheme. System throughput has been defined as the amount of messages

that can be handled by an application software system within a unit of time [6]. The message from one node is estimated to be 1 byte, the messages increase as the number of nodes varies. The mathematical model for this experiment was developed in Section 3. It is clear from Eq. (13) that as the number of messages successively delivered increases within the simulation time, the system throughput increases accordingly as shown in Figure 4. In the graphical analysis the poor performance of RPC is very clear when compared with mobile agent and lightweight agents. Lightweight agents show a superior performance against single mobile agent in information delivery scenario. Generally, the throughput of lightweight agents is double that of RPC while it is significantly higher than single mobile agent throughput. It is interesting to note that throughput decreases as the load across the network increases.

7.3. System latency

System latency is the time required by software application to successfully process a piece of information and send report back to the user. If the system requires long time to process information the real-time result becomes too long and the information may not be available in timely fashion.

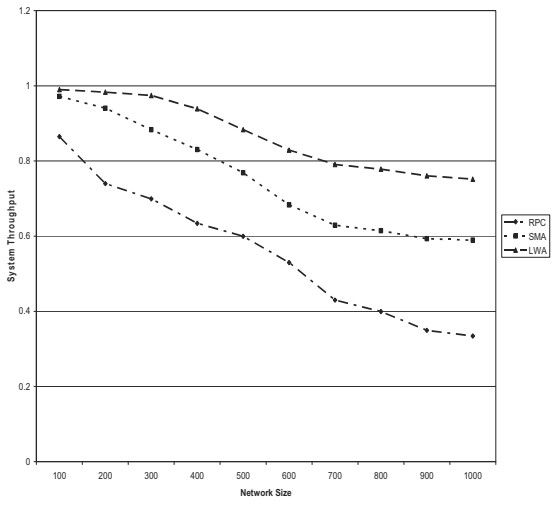


Figure 4. Throughputs of the three schemes.

This is important when applications are used in life saving processes or devices, top management decision taking or policy, etc. In this experiment the latency of each scheme is compared, and from Figure 5, it is shown that lightweight agents have the least latency. The latency of RPC is so high because the communication mechanism of RPC requires the scheme to use the network resources (bandwidth) throughout its computation. The network failure does not favor long use of bandwidth as it leads to retransmission of information to complete a service. This increases the latency of the scheme. Single mobile agent has poor performance when compared with Lightweight agents because the load for a single agent is shared among many agents, this makes the processing faster. For the three schemes, as the load across the network increases the latency increases this shows that more time is required for the system to process the information.

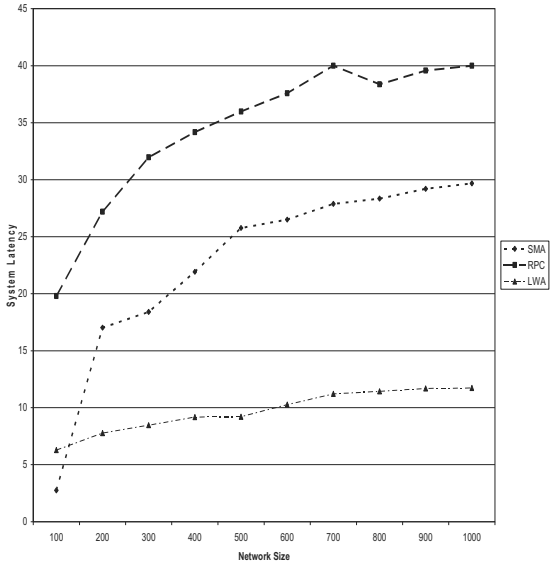


Figure 5. Delays of the three systems.

8. Conclusion

In this paper, we have carried out comparative evaluation of LWAs, SMA and RPC based on system overhead with retransmission, system throughput and system latency in distributed computing environment. Our evaluation is based on the simulation results of the schemes. The results of the simulation on each of the performance metrics show that lightweight agents are well favored for data communication in distributed computing network than the other two schemes. It is shown that as the number of

network elements increases on the network, the three schemes experience performance deterioration, but the RPC case is worst due to its proportionate increase which makes it more liable to network failure. The simulated network failure was fixed at 15% for the simulation. Generally, lightweight agents prove to be more efficient and fault tolerant than the other two existing schemes.

References

[1] G. VOLPATO, A. STOCCHETTI, The role of ICT in the strategic integration of the automotive supply chain. *Journal of Automotive Technology and Management*, (2002) Vol. 2, pp. 239–260.

[2] R. A. VASUDEVAN, A. ABRAHAM, S. SANYAL, D. AGRAWAL, Jigsaw-based Secure Data Transfer over Computer Networks. *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)* (2004) Vol. 2, pp. 2–11.

[3] T. WHITE, B. PAGUREK, Towards Multi Agent Problem Solving in Networks. *Proceedings Third International Conference on Multi-Agent Systems (ICMAS'98)*, (1998) pp. 333–340.

[4] G. H. JOHNNY, S. K. WONG, V. HONAVAR, L. MILLER, Y. WANG, Lightweight Agents for Intrusiondetection. *The Journal of Systems and Software*, 67 (2003), pp. 109–122.

[5] G. A. ADEROUNMU, Performance Comparison of remote procedure call and mobile agent approach to control and data transfer in distributed computing environment. *Journal of Network and Computer Applications*, 27 (2004), pp. 113–129.

[6] M. SAMARAH, P. CHAN, Enabling Mobile Agents Communication. *In the proceedings of Fourth Rin International Workshop on Multi-agents*, (2001) available at: `https://www.cs.fit.edu/Projects/tech_reports/cs-2000-3.pdf` [Feb, 15, 2007].

[7] F. DOUGLAS, D. MILOJICIC, Y. PAINDAVEINE, R. WHEELER, S. ZHOU, Process Migration. In *ACM Computing Surveys*, 32(3) (2000), pp. 241–299.

[8] A. DELIS, N. ROUSSOPOULOS, Performance Comparison of Three Modern DBMS Architectures. *IEEE Transactions on Software Engineering*, Vol. 19(2) (1993), pp. 120–138.

[9] E. A. OLAJUBU, G. A. ADEROUNMU, E. R. ADAGUNODO, Optimizing Bandwidth usage and Response Time using Lightweight Agents on Data Communication Network. T. Sobh et al. (eds), *Novel Algorithms and Techniques in Telecommunications, Automation and Industrial Electronics*, (2008) pp. 335–340.

[10] M. BERNA-KOES, I. NOURBAKHSH, K. SYCARA K., Communication Efficiency in Multi-Agent System. *In Proceedings of International Conference on Robotics and Automation*, (2004) pp. 2129–2134.

[11] V. BAUSIS, S. HADJIEFTHYMIADES, G. ALYFANTS, L. MERAKOS, Autonomous Mobile Agent routing for efficient server resource allocation. *Journal of Systems and Software*, 82(5) (2009), pp. 891–906.

[12] S. VENKATESAN, C. CHALLAPPAN, Generating Routing Table for Free roaming Mobile Agent in Distributed Environment. *Computer Standard & Interfaces* 31(2) (2009), pp. 428–436.

[13] D. RUTTER, A Performance Comparison of Mobile Agents and RPC. *Lecture Notes in Computer Science*, (1999), pp. 441–448.

[14] C. IMREH, T. NÉMETH, On Time Lookahead Algorithms for the Online Data Acknowledgement L. Kučera and A. Kučera (Eds.): MFCS 2007, LNCS 4708, pp. 288–297, Springer-Verlag, Berlin, Heidelberg.

Received: November, 2008
Accepted: April, 2011

Contact addresses:
Emmanuel A. Olajubu
Computer Building Complex (Room 104)
Department of Computer Science & Engineering
Obafemi Awolowo University
Ile-Ife, Nigeria
e-mail: `emmolajubu@oauife.edu.ng`

Ganiyu A. Aderounmu
Department of Computer Science & Engineering
Obafemi Awolowo University
Ile-Ife, Nigeria
e-mail: `aaderoun@oauife.edu.ng`

EMMANUEL A. OLAJUBU is a faculty member in the Department of Computer Science & Engineering, Obafemi Awolowo University, Ile-Ife, where he graduated in 1997. He holds research degrees M.Sc. and Ph.D. in computer science obtained from the same department (2003 and 2008 respectively). He is also a member of Nigerian Computer Society (NCS), Computer Professional Registration Council of Nigeria (CPN) and other international professional bodies. He has over 10 years experience in teaching and research. He has a number of scholar scientific publications in journals of international repute and conference proceedings both at national and international level. His research interest includes computer communication and network, mobile agent systems and soft computing.

GANIYU A. ADEROUNMU holds a research degree M.Sc./Ph.D. in computer science obtained from Obafemi Awolowo University, Ile-Ife, Nigeria (1997 and 2001 respectively). He is a member of the Nigeria Society of Engineers (NSE) and also a registered computer engineer with the Council for 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 15 years of experience in teaching and research. He is an author of many journal articles in Nigeria and abroad. His special interest includes computer communication and network. He is a Visiting Research Fellow to the University of Zululand, Republic of South Africa. He is an associate professor in the Department of Computer Science & Engineering. He was the former head of the Department of Computer Science & Engineering and now the Acting Director of Information Technology and Communication Unit (INTECU).

ANUOLUWAPO O. AJAYI holds a Ph.D. degree in computer science obtained from the Obafemi Awolowo University (OAU), Ile-Ife, Nigeria. He is a lecturer at the department of Computer Science and Engineering, OAU, Ile-Ife. His research interests include artificial intelligence and programming languages techniques. He is a member of International Association of Engineers (IAENG), Nigerian Computer Society (NCS) and IEEE Computer Society. He has published scientific articles in several journals of international repute.