Network resources management in a multi-agent system: A simulative approach

Authors:
Emmanuel A. Olajubu[1]
Ganiyu A. Aderounmu[1,2]
Emmanuel R. Adagunodo[1]

Affiliations:
[1]Department of Computer Science and Engineering, Obafemi Awolowo University, Nigeria

[2]Department of Computer Science, University of Zululand, South Africa

Correspondence to:
Emmanuel Olajubu

email:
emmolajubu@oauife.edu.ng

Postal address:
PO Box 3352, Agege Post Office, Agege, Lagos 234, Nigeria

Keywords:
bandwidth usage; Bernoulli random variable; lightweight agents; multi-agent; throughput

Dates:
Received: 04 May 2009
Accepted: 30 Apr. 2010
Published: 20 Sept. 2010

How to cite this article:
Olajubu EA, Aderounmu GA, Adagunodo ER. Network resources management in a multi-agent system: A simulative approach. S Afr J Sci. 2010;106(9/10), Art. #322, 6 pages. DOI: 10.4102/sajs.v106i9/10.322

This article is available at:
http://www.sajs.co.za

ABSTRACT

Multi-agent systems (i.e. systems comprising many agents) have been proposed for many Internet and distributed applications. The proposed systems have little or no consideration of the effects of this multi-agent approach on network resources. In this paper, we presented a simulation assessment of the effect of multi-agent systems on network resources. The routing scheme of the agents was formulated based on the travelling salesman problem. Lightweight agent (LWA) controller was modelled using a fuzzy logic toolbox in the MATLAB environment. The performance metrics of bandwidth usage, response time and throughput were used to compare the network resources usage by different groups of LWAs (10 LWAs, 40 LWAs, 100 LWAs, 150 LWAs) during their computational task on the network. Java programs were written for the implementation of lightweight agents in the simulation. The inputs to the system were realised by multiplicative pseudorandom number generation during the simulation. The simulation result analysis was carried out based on the performance metrics stated above for the four groups of agents. Increasing the number of LWAs in a simulated multi-agent system decreased the response time but increased the throughput and the bandwidth usage. All these performance measures should be considered for developing countries with bandwidth shortages, because having too many agents in a multi-agent system could result in bandwidth wastages.

INTRODUCTION

An aggressive demand for the enormous services and information provided by the Internet presupposes economic conditions that can guarantee good support for the ubiquitous network. Several means have been sought to ensure connectivity to end users at a reduced cost. As a result of economic pressure, it is clear that several users will still connect to the Internet with low-bandwidth connection devices. Although Internet traffic is growing rapidly, especially in developing economies, the provision of bandwidth for Internet connections through the laying of an infrastructural Internet backbone in many of these nations has been narrow. Bandwidth availability for many of these end users will therefore remain inadequate because of several technological factors and some end users will still be forced to connect via dial-up modems, or, at best, ADSL over the old copper wire system. Many other users will connect via low-bandwidth wireless networks. Most rural communities of developing nations cannot afford to make more than 128 Kbps – 1 Mbps bandwidth available to their institutions and establishments.[1] To meet the demands for Internet through low-bandwidth connections, mobile agents have begun to offer a better solution than existing technologies.[2]

Owing to their flexibility, efficiency, reduced bandwidth usage and fault tolerance, researchers have proposed multi-agent systems (i.e. systems comprising many agents; MASs) for data transmission on computer networks.[3] The proposed multi-agent systems, however, do not consider the available bandwidth on a computer network. As mobile agents are codes that make use of bandwidth during their itinerancy, multi-agent systems will improve network bandwidth usage during their communication and will also develop cost-effective path planning for a mobile agent system without any emphasis on the implication of the agents on network resources, though it was noted that bandwidth usage in a MAS may be directly proportional to the number of agents in the system without experimental confirmation. A reasonable amount of research has also been done to investigate the degree of network resources usage between mobile-agent technology and remote procedure call protocols.[2] The bases of comparison were bandwidth usage, throughput and response time. Furthermore, remote procedure call protocols and mobile agents have been compared on the basis of their scalability.[5]

To the best of our knowledge, no studies have considered the impact of MASs on network infrastructures. Increasing the number of agents in MASs without due consideration for bandwidth, may erode the acclaimed low-bandwidth usage of mobile-agent technology. To be able to showcase agents' resources consumption on a computer network, we have varied the number of agents, so as to show the gap in resources used by the different group. In line with the above, this paper presents a simulation assessment of the effect of multi-agents on network resources, especially network bandwidth. This work is aimed at researchers and software developers who have a special interest in the development and deployment of MASs of some important issues to take into consideration during software development. We discuss multi-agent systems in the following section, after which we present the model analysis of the proposed system. We then briefly discuss the agents' routing scheme and model formulation and present the simulation experiment for the system and its results, before offering our conclusions.

MULTI-AGENT SYSTEMS

Mobile-agent technology has been widely accepted among computer science researchers and network-centric application developers due to some salient advantages this technology promises to offer over the traditional client-server architecture. The wide acceptance of the technology is partly because of its fault-tolerance during network downtime and reduced latency (and therefore optimum bandwidth usage) on a computer network.[1] The advantages and potential of agent-to-agent communication and

South African Journal of Science
Article #322

agent-to-other-entity communication has greatly fuelled MAS research, situating MASs as a tool to solve many complex Internet applications.[4,6] Some authors, such as Abrahams and Dai[7] and Espinasse et al.[8], proposed a MASs architecture for information storage, search and retrieval on the Web. In the same manner, other studies[9,10,11] aimed at solving complex problems of distributed diagnosis and patient monitoring over the Internet through MASs. Mandureira et al.[12] opined that MASs applications are the appropriate technology to use for removing the complexity in network scheduling, especially in the computer networks of manufacturing companies.

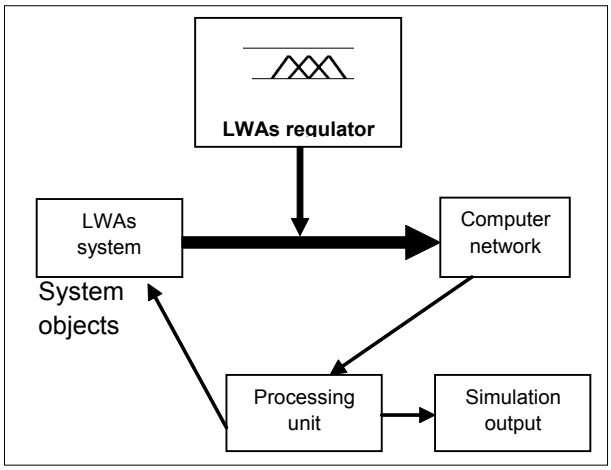
The number of researchers who propose to solve complex network applications through MASs has increased greatly, with little or no consideration of how the MASs will impact on network bandwidth. Irrespective of this claim, mobile agents are program codes that make use of bandwidth during their itinerancy. The amount of bandwidth usage is a function of the number of agents in the systems[4,13] and so it is necessary to know the implications of having many agents in a system so as to be able to strike a balance between the numbers of agents in a MAS and the available bandwidth. According to Oluwatope et al.[14], bandwidth provisioning without effective management will not solve bandwidth scarcity. Optimal usage of available bandwidth resources on any network is likely to increase network quality of service (QoS) for the end user. To this end, we assess the impact of many agents within a system on computer network resources in this paper.

Structure of lightweight agents

The present ubiquitous networks are usually conglomerates of many heterogeneous, very often incompatible, multi-vendor components with different operating systems. The networks that usually have many computer systems, each with a different operating system, require a robust and flexible scheme, which can perform computational tasks among the various data sources and different operating systems on a distributed system that may span geographical distance. When a single agent is created to transmit data among all of these nodes with different operating systems, it is incumbent on the agent to carry all the intelligence required to communicate and perform computational tasks across operating system platforms, which makes the agent too large in code size. This form of agent is referred to as a heavyweight agent,[15] which does not use network resources optimally during its itinerancy. Thus it is necessary to design an agent that is small in code size (lightweight) and which has the following characteristics:

- a focus on minimalism
- the functionality of being easily programmable
- the ability to transmit data rapidly
- the adaptability of being dynamically updatable and upgradable.

Lightweight agents (LWAs) allow dynamic aggregation,[15] which enables the addition of new run time capabilities. Collaborative facilities can be added to the lightweight mobile agents, which can quickly add new features to the system. Since storage facilities are no longer issues in computing, different types of intelligence needed by the agents are stored in various systems where agents are required to carry out computational tasks. The agents simply migrate to the system and download the required intelligence, thereby upgrading themselves for the computational task. After completing the work on the system, the downloaded intelligence is discarded before the agent migrates to another system. This underscores the principle of minimalism. In essence, there are three components that are basic to every mobile agent, (1) the code is a computer program (written in object-oriented language) that specifies the mobile agent's characteristics, (2) attributes describe the mobile agent, its origin and owner, its itinerary history and the resources required for execution, authentication et cetera, all of which usually may not be modified by the mobile agents themselves and (3) an execution state describes the mobile agent's internal variables that enable it to terminate its execution on a host and to migrate through a computer network to another host, before resuming its execution.



LWAs, lightweight agents.

FIGURE 1
Simulation architecture

MODEL ANALYSIS

A comparative simulation assessment technique was carried out for the group of agents used in this simulation. The simulation architecture is depicted in Figure 1. This architecture consists of five modules, namely, (1) a computer network, (2) system objects such as the LWAs, (3) a processing unit, (4) an output unit and (5) the LWA regulator.

Computer network

The computer network used in this simulation is a virtual network, where the four groups of agents carry out a computational task, namely information retrieval from the different nodes the agents visit. All the nodes on the network are enabled for mobile agents, that is, mobile agents are allowed to perform their computational task on the nodes. Each computer or computational entity on the network has a varying processing capacity and each system is autonomous. The system must have an agent-enabled platform before it can accept and run any agent.

The system objects

The system objects are the four groups of lightweight agents (10 LWAs, 40 LWAs, 100 LWAs and 150 LWAs), the impact of which was to be assessed on the network. The agents in each group migrate from the home node (server) where they reside to the computational entities (nodes) on the network for information retrieval activities. After the computational task, the agents are expected to return to the home node, completing the roundtrip of the agents' itinerary.

LWAs regulator

During simulation, the LWA regulator controls the number of LWAs released onto the network, in order to assess the impact of each group.[16] To regulate the agents, this object uses the fundamentals of control theory, which is a well-established branch of engineering and offers a range of powerful techniques for use in designing and analysing complex systems. Such systems (e.g. an LWA regulator) are easily controlled; for instance, it is possible to control parameter X by adjusting variable Y. The self-regulating power of the lightweight agents controller (LWAC) depends on the amount of available bandwidth and the number of nodes (network size) to determine the number of agents that will be released to the network for a computational task. As the available bandwidth increases and the number of nodes on the network increases, more agents are

Article #322 South African Journal of Science

released to the network by the LWAC. If either bandwidth or the network size is kept constant, the number of agents that will be sent to the network will not be in the same proportion as if the two parameters both increase.

The LWAC is a rule-based system that uses the 'if ... then' criterion to implement its knowledge acquisition. The two variables that the system uses to determine the number of LWAs to be released to any network are, (1) the number of nodes on the network, that is, the network size (NN) and (2) the available bandwidth (ABD). The two variables are the set of inputs supplied by the network administrator to the controller so that the optimal number of LWAs can be generated for the specified network.

Fuzzy rule format for the LWAs regulator knowledge base

The rule-based system for the controller uses two input variables and one output variable as both the conditions and the conclusion of the rules. The multi-input–single-output (MISO) applies to this controller.

There are 56 rules in the knowledge base of the controller. The 'if … then' rule statement is used to formulate the conditional statements that comprise the knowledge base by assuming the form: 'if Y is k, then Z is l'. The 'if' part of the rule is known as the premise and the 'then' is the consequence. The rule base of this system makes use of a forward-chaining system. The forward-chaining system processes the initial fact first, after which the rules are used to draw a conclusion based on the processed data. The forward-chaining system is thus said to be data driven. Samples of the applied rules are shown below; these are merely samples and not the whole rules used in developing the system:

- If NN is VS and ABD is VP then NAG is VA.
- If NN is VS and ABD is LR then NAG is FA.
- If NN is ST and ABD is PR then NAG is VA.
- If NN is AT and ABD is FR then NAG is SA.
- If NN is FL and ABD is MB then NAG is LV.

The consequence is the number of agents (NAG) that will be released to the network. Table 1 shows the data fuzzification for inputs and output of the system and is divided into three parts – A, B and C – to reflect the different variables used and their linguistic meaning. Table 1A and 1B are the fuzzification of the two inputs, while Table 1C is the fuzzification of the output. The few rules showcased in this paper are not the rules that were used in the simulation, but we have 56 rules in the knowledge base of the system. The knowledge base is interfaced with our simulation program to produce the results we discuss below.

TABLE 1
Controller input and output data fuzzification

Table 1A: Available bandwidth

Fuzzy logic variable	Linguistic variable	Available bandwidth
VP	Very poor bandwidth	0.1–1.0
PR	Poor bandwidth	0.8–1.5
FR	Narrow bandwidth	1.1–1.3
GD	Fair bandwidth	2.5–4.5
LR	Good bandwidth	3.5–6.0
VL	Large bandwidth	5.0–7.0
MX	Very large bandwidth	6.5–8.5
MB	Maximum bandwidth	8.0–10.0

Table 1B: Number of nodes on the network

Fuzzy logic variable	Linguistic variable	Number of nodes
VS	Very small network	20–75
ST	Small network	50–150
AT	Average network	100–250
FL	Fairly large network	180–350
LT	Large network	300–450
VL	Very large network	400–600
EL	Extra large network	530–750
IT	Increased network size	680–880
MT	Maximum network	820–1000

Table 1C: Number of agents

Fuzzy logic variable	Linguistic variable	Number of agents
VA	Very small number of agents	1–3
SA	Small number of agents	2–5
AV	Average number of agents	4–8
FA	Fairly large number of agents	6–10
LA	Large number of agents	9–13
IA	Increased large number of agents	11–15
LV	Very large number of agents	14–18
MA	Maximum number of agents	17–20

Processing unit

The processing unit is the object where the processing of the data takes place. This object also sends the result to the output unit.

Output unit

The output unit presents the results of the simulation in a numeric form to the user. The architecture shown in Figure 1 functions in the following way: the four groups of agents migrate to the network for the computational task and return to the home node where the scheme resides. This process normally can proceed uninterrupted if network failure does not occur. Network failure, however, is a common phenomenon in communication networks and will interrupt the process and possibly necessitate the retransmission of data in order to complete the service. The network failure is generated using a Bernoulli random variable generator.

ROUTING SCHEME AND MODEL FORMULATION

The routing scheme is conceptualised as a travelling salesman problem (TSP) scheme, which is formulated as follows: a mobile agent wishes to visit an N distinct nodes on the network and return to the home node. The latency between node k and node $k + 1$ 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 assignment is to find the sequence of tours, such that the overall distance travelled or cost is minimised. In one itinerary of the agent, all the available nodes on the network are visited once. Multiple visits to a node in one itinerary are not allowed in the TSP concept. The TSP itinerary for mobile agents in this system is given in [Eqn 1], which depicts the optimisation problem.

$$\text{Min. } Z = \sum_{k=1}^{N} (t^l_{k,k+1} + t^c_k) X_{k,k+1} \qquad \text{[Eqn 1]}$$

In [Eqn 1], the agents want to visit N nodes on the network such that all nodes are traversed once in one itinerary. The problem is that the agents should find an optimal path that will minimise the cost (Z) of their itinerary. The latency between nodes k and $k + 1$ is denoted by t^l, while t^c is the computational cost on node k, which is subject to the calculation in [Eqn 2]:

$$X_{k,k+1} \in \{1,0\} \text{ for all } k, \text{ where } k = 1,2...m \qquad \text{[Eqn 2]}$$

The artificial variable X in [Eqn 2] removes all sub-tours on a node and assumes the values of 1 or 0. When it assumes value 1, all other sub-tours on the nodes assume 0, which indicates that those paths are not feasible.

There are at least two sub-paths on each node. The most viable path to take among these sub-paths is a question of the agent's intelligence. In this work we have adopted the shortest path algorithm, as described by Chang and Zhang[17], to solve this problem. The algorithm processes the nodes on the network piece by piece (i.e. each node is processed one at a time). The agent on the current node calculates the shortest possible route to the next adjacent node on the network. The agent moves there and repeats the same exercise until all the nodes are visited. From our hypothetical network presented in Figure 2, the agent started its itinerary from PC1, which was assumed to be our

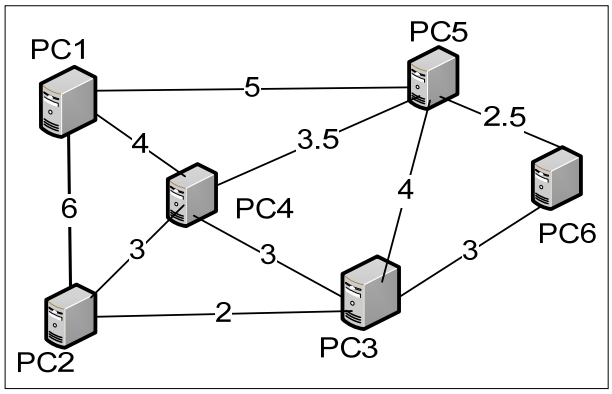


FIGURE 2
Agent routing scheme

source node. It then looked for the shortest path to the next node (PC4), from which it repeated the algorithm to get to PC3. The exercise continued until all the nodes were visited. On PC5 it was not possible for the agent to move back to PC3 or PC4 because those two nodes were already visited; the agent had to migrate to the source node (PC1). Scalability is not a problem in a mobile-agents system, so, as the network grows in number of nodes, the agents were still able to find the optimal path, because each node on the network was processed individually.

The sub-paths available on each node on the network are shown in Table 2. These sub-paths form the various alternative paths that agents can take on each node to continue their itinerary. The agents select the most optimal path using shortest path algorithm.

Our TSP only allows a node to be visited once in a single itinerary. In the six nodes presented as example, the optimal path of the agents is: PC1 → PC4 → PC2 → PC3 → PC6 → PC5 → PC1.

To formulate the asymmetric TSP for a network of N nodes, there is the need to introduce an artificial variable X, which is a zero-one variable that eliminates the sub-tours on each node on the network.

The agent moves on the network, with its code size X_{lwa}, and returns with the same code size to the source node, therefore, the size for the complete journey is assumed to be $2X_{\text{lwa}}$ and the number of nodes in the network is N. It is assumed that a query y is negligible when compared with the agent's code size, therefore it is assumed to be 0, while the agents' computational result, otherwise known as the reply, is represented as z bytes; the number of nodes on the network is N, where N ranges from $j = 1, 2 \ldots N$, the last node on the network. Then the total bandwidth usage $C_{\text{bandwidth}}$ for LWAs is given by [Eqn 3]:

$$C_{\text{bandwidth}} = \sum_{j=1}^{N} \sum_{i=1}^{K} (2X_{\text{lwa}} + z)_i \qquad \text{[Eqn 3]}$$

where K is number of agents involved in computational tasks in the network and $i = 1, 2, 3 \ldots K$.

TABLE 2
Sub-paths available on each node on the network

Node	Available sub-paths
PC1	PC1 → PC5; PC1 → PC4; PC1 → PC2
PC2	PC2 → PC1; PC2 → PC4; PC2 → PC3
PC3	PC3 → PC2; PC3 → PC4; PC3 → PC6; PC3 → PC5
PC4	PC4 → PC1; PC4 → PC2; PC4 → PC3; PC4 → PC5
PC5	PC5 → PC1; PC5 → PC3; PC5 → PC4; PC5 → PC6
PC6	PC6 → PC3; PC6 → PC5

SIMULATION EXPERIMENT

Simulation environment description

The MATLAB program is a popular tool for designing computer models both in industry and academia, with applications for both science and engineering. Researchers have employed it to model many systems, especially those used in information technology. MATLAB is regarded as a technical high-performance tool for computing with many examples, instructions and apparatus for designing applications and evolving computer algorithms. In this research, the fuzzy logic model was developed in the MATLAB fuzzy logic toolbox, while the developed model was interfaced with the Java program developed for LWAs. MATLAB does not yet use an object-oriented language and therefore could not be used to write the mobile-agent program.

Description of the simulation parameters

The simulated experiment allocates agents to a network in a multi-agent system based on the available bandwidth (AB) and network size (NN). Three experiments were conducted for the simulation. Our simulation experiment was carried out on a Pentium 2.8 GHz CPU, with 512 MB of RAM, running on the Windows XP operating system. The two inputs used a random number generator to determine the inputs to be fed into the system during simulation; the upper band for the input NN was 1000, while the lower band was 100. Also, the input AB had an upper band of 10 Mbps and a lower band of 0.1 Mbps.

Tiny mobile agents for this work were developed in the Java programming language environment and interfaced with the fuzzy model developed for experimental purposes. A dynamic virtual network, the network elements of which ranged between 100 and 1000 nodes, was set up in Java run time environment for the purpose of the simulation. The network allowed the dynamism of the modern network as mobile devices could easily come into the network and leave at any time. As discussed in the model analysis section, the input data to the LWAC, the number of nodes and available bandwidth, and the output data (the number of agents) were fuzzified (Table 1). A number of agents were released to the network to investigate the impact of agents on the network resources usage.

SIMULATION RESULTS AND DISCUSSION

Bandwidth usage

The bandwidth usage shown here represents the amount of data transmitted across the network by the different number of LWAs that traverse the network at a unit time. The agents traverse the network with their code and then download the required intelligence on the destination node for the specific task it is assigned to accomplish on the node. Factors that contribute to the bandwidth used by an agent are the agent's code and the technology employed (the intelligence of the agent) to perform its assigned task. It is clear from Figure 3 that the bandwidth used is directly proportional to the number of agents (multi-agents) on the network for remote computational work. It is obvious that, as the number of agents on the network increases, the bandwidth used increases. The following numbers of agents (10 LWAs, 40 LWAs, 100 LWAs and 150 LWAs) were allowed onto the network for the purposes of our simulation. The result was used to present the degree of bandwidth usage as the network size varied. The bandwidth consumption of 150 LWAs was the highest amongst all the groups of LWAs, while 10 LWAs had the lowest bandwidth consumption; these results indicate that as the number of agents in a system increases, the network resources usage (in terms of bandwidth usage) increases proportionally and are shown in Figure 3 and Table 3.

Response time

Response time is defined as the period of time required for the processing of information and the return of a result to the end user. This time is often conceptualised as a delay in information systems. In this work, 'division of labour' is employed by the

Article #322 South African Journal of Science

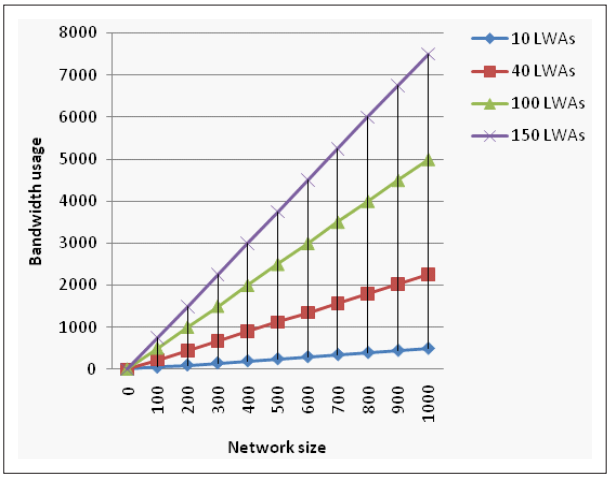


FIGURE 3
Bandwidth usage (kbps) of groups of lightweight agents (LWAs) when network size is increased

TABLE 3
Simulated bandwidth usage (kbps) of groups of lightweight agents (LWAs) when network size is increased

Network size	10 LWAs	40 LWAs	100 LWAs	150 LWAs
100	50	225	500	750
200	100	450	1000	1500
300	150	675	1500	2250
400	200	900	2000	3000
500	250	1125	2500	3750
600	300	1350	3000	4500
700	350	1575	3500	5250
800	400	1800	4000	6000
900	450	2025	4500	6750
1000	500	2250	5000	7500

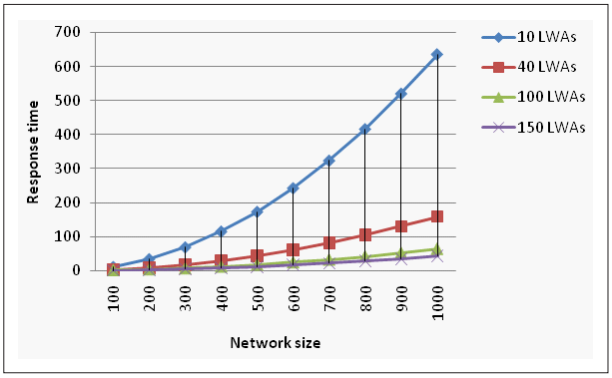


FIGURE 4
System response time (s) of groups of lightweight agents (LWAs) when network size is increased

TABLE 4
System response time (s) of groups of lightweight agents (LWAs) when network size is increased

Network size	10 LWAs	40 LWAs	100 LWAs	150 LWAs
100	11.55	2.89	1.16	0.77
200	34.65	8.66	3.47	2.31
300	69.30	17.33	6.93	4.62
400	115.50	28.88	11.55	7.70
500	173.23	43.31	17.33	11.55
600	242.55	60.64	24.26	16.17
700	323.40	80.85	32.34	21.56
800	415.80	103.95	41.58	27.72
900	519.75	129.94	51.98	34.65
1000	635.25	158.81	63.53	42.35

system to share a computational task among all of the agents assigned to it; this allows for the equal sharing of network nodes among all participating agents. The amount of data processed by an individual agent is a function of the number of agents involved. The individual agent moves around on the network for information search and filtering. Thus, fewer nodes are required to process the data when more agents are involved, which is why the simulation result for this experiment reflected a very good response time from the largest group of agents. However, this good response time was achieved at the expense of network bandwidth, because increasing the number of agents increased the required bandwidth.

Figure 4 shows that the response time of 10 LWAs was very low, indicating that individual agents in the group had a greater number of nodes to visit and a larger volume of data from the system components to process; therefore, it took a longer time before information could reach the end user. When a limited number of agents is used for a computational task, the end users will have to pay more (i.e. increasing the transaction length necessitates that more resources, such as bandwidth, are consumed), because response time is going to be very low. This data can be contrasted to that produced from the use of 150 LWAs, where individual agents in the group had fewer nodes to visit and, consequently, could process the information faster than the 10 LWAs group. It is clear that using 150 LWAs will result in a shorter response time, but it will also mean the Internet Service Provider (ISP) will have to pay more in terms of bandwidth provisioning. The end user will prefer high QoS in terms of response time and thus may be attracted to this network (150 LWAs), rather than where 10 LWAs route the network. The faster response of 150 LWAs, compared with 10 LWAs or 40 LWAs (Table 4) is at the expense of bandwidth.

The small differences in response times between the simulation results for 100 LWAs and 150 LWAs showed that overly increasing the number of agents monitoring the network elements in order to achieve a higher response time, will result in unnecessarily increasing the bandwidth, because, at a point, the increase in the number of agents will not justify the increase in the corresponding response time.

System throughput

Information system (software) throughput is generally defined as the volume of data the information system can process within a unit of time. Also, throughput is defined as the number of messages the system can handle within a sample interval of time.[18] We have adopted the second definition for this work.

The system throughput defines the performance of the four groups of lightweight agents (10 LWAs, 40 LWAs, 100 LWAs and 150 LWAs) in terms of volume of data transmitted across the network within a unit of time. It is assumed that there is intermittent network failure due to network congestion. This assumption affected the four groups of LWAs differently and these differences are visible in the simulation result. In the simulation, we assumed that a 20% volume of processed data would be retransmitted and thus we deliberately ignored cases where there was a network outage due to the breakdown of network elements, because there was no transmission in these periods.

As the number of agents in the simulation increased, the data to process by individual agents decreased. For a system with fewer agents, but the same number of network nodes, retransmission due to congestion is likely going to increase and, therefore, reduce the throughput. However, when there were more agents (e.g. 100 LWAs or 150 LWAs, see Table 5), the amount of information to be processed was less for individual agents, so the network congestion effect was thus likely to be minimal, which is why the throughput for 150 LWAs was far higher than the throughput for 10 LWAs. Figure 5 graphically depicts the system throughput of the simulation.

South African Journal of Science Article #322

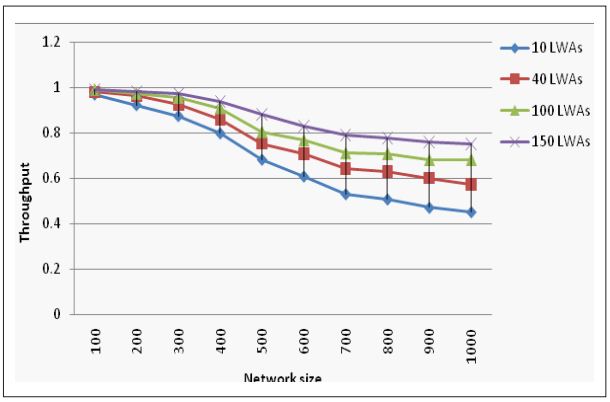


FIGURE 5
System throughput of groups of lightweight agents (LWAs) when network size is increased

TABLE 5
System throughput of groups of lightweight agents (LWAs) when network size is increased

Network size	10 LWAs	40 LWAs	100 LWAs	150 LWAs
100	0.970	0.980	0.990	0.990
200	0.923	0.963	0.973	0.983
300	0.875	0.925	0.955	0.975
400	0.799	0.859	0.909	0.939
500	0.683	0.753	0.804	0.883
600	0.609	0.709	0.767	0.829
700	0.531	0.641	0.711	0.791
800	0.508	0.628	0.708	0.778
900	0.471	0.601	0.681	0.761
1000	0.452	0.572	0.682	0.752

CONCLUSION

Multi-agent systems are a good tool for implementing distributed system applications and many application developers are willing to explore the advantages of multi-agent systems for Internet software applications, especially in a low-bandwidth network. Nevertheless, maximisation of the profits of a multi-agent system may lead to a situation of having too many agents in a system, which would lead to bandwidth wastages. In this paper, we carried out a comparative analysis of many mobile agents that route a computer network using the TSP scheme on network resources. The LWAs in the system were regulated by the LWAC, which is rule-based. The analysis and the results of this work are based on a simulation rather than a practical implementation of the system. Therefore, further work is required in order to determine how the implementation of the system will improve QoS on communication networks in developing nations in which there is an acute shortage of bandwidth provisioning.

ACKNOWLEDGEMENTS

We are sincerely grateful to the Academic Staff Union of Universities (ASUU) for their support for this work through the ASUU research grant award of 2007. It should be noted that the views expressed in this paper are those of the authors and not ASUU.

REFERENCES

1. Kotz D, Gray RS. Mobile agents and the future of the Internet. ACM SIGOPS – Operating Systems Review. 1999;33:7–13.
2. Aderounmu GA. Performance comparison of remote procedure calling and mobile agent approach to control and data transfer in distributed computing environment. J Netw Comput Appl. 2004;27:113–129.
3. Sygkouna T, Anagnostou M. Efficient information retrieval using mobile agents. Paper presented at: AAMAS 2005. Proceedings of the 4th International Joint Conference on Autonomous Agents and Multi-agent Systems; 2005 July 25–29; Utrecht, the Netherlands. New York: ACM; 2005. p. 1241–1242.
4. Baek J, Kim J, Yeom H. Timed mobile agent planning for distributed information retrieval. Paper presented at: the 5th International Conference on Autonomous Agents. Proceedings of the Fifth International Conference on Autonomous Agents; 2001 May 28–June 01; Montreal, Canada. New York: ACM; 2001. p. 120–121.
5. Rutter D. A performance comparison of mobile agents and RPC. Lecture notes in computer science, vol. 1725. Berlin: Springer-Verlag, 1999; p. 441–448.
6. Jennings NR. An agent-based approach for building complex software systems. Comm ACM. 2001;44(4):35–41.
7. Abrahams B, Dai W. Architecture for automated annotation and ontology based querying of semantic web resources. Paper presented at: IEEE/WIC/ACM 2005. Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence; 2005 Sep 19–22; Compiègne, France. Washington DC: IEEE Computer Society; 2005. p. 413–417.
8. Espinasse B, Fournier S, Freitas F. Agents and ontology-based information gathering on restricted Web domain with AGATHE. Paper presented at: ACM SAC 2008. Proceedings of the 2008 ACM Symposium on Applied Computing; 2008 Mar 16–20; Fortaleza Ceara, Brazil. New York: ACM; 2008. p. 2381–2386.
9. Webber CG, Silva JLT. Self and non-self discrimination agents. Paper presented at: the 2008 ACM Symposium on Applied Computing. Proceedings of the 2008 ACM Symposium on Applied Computing; 2008 Mar 16–20; Fortaleza Ceara, Brazil. New York: ACM; 2008. p. 1987–1988.
10. Mabry S, Eetters T, Schneringer T, Edwards N, Hug C. Addressing the complexity of patient monitoring with multi-agent and modular logic. Paper presented at: AAMAS 2003. Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-agent Systems; 2003 July 14–18; Melbourne, Australia. New York: ACM; 2003. p. 1058–1059.
11. Ghosh S, Qutaiba Razouqi H, Schumacher J, Celmins A. A survey of recent advances in fuzzy logic in telecommunications networks and new challenges. IEEE Trans Fuzzy Systems. 1998;6(3):443–447.
12. Mandureira A, Santos F, Preveria I. Self managing agents for dynamic scheduling in manufacturing. Paper presented at: GECCO. Proceedings of the 2008 GECCO Conference Companion on Genetic and Evolutionary Computation; 2008 July 12–16; Atlanta, USA. New York: ACM; 2008. p. 2187–2192.
13. Baek J, Kim G, Yeo J, Yeom H. Cost effective mobile agent planning for distributed information retrieval. Paper presented at: ICDCS-21 2001. Proceedings of the 21st International Conference on Distributed Computing Systems; 2001 April 16–19; Las Vegas, USA. Washington DC: IEEE Computer Society; 2001. p. 65–72.
14. Oluwatope AO, Aderounmu GA, Adagunodo ER, Akinde AD. Stochastic reward net end-to-end quality of service (QoS) simulation modeling across ATM networks: Using the goodput model. Paper presented at: IEE QoS 2004. Proceedings of the IEE Telecommunication and Quality of Service: The Business of Success conference; 2004 Mar 02–03; London, UK. Earls Barton: Wrightsons Ltd.; 2004. p. 165–170.
15. Wooldridge M. An introduction to multi-agent systems. Indianapolis: John Wiley & Sons Ltd.; 2002.
16. Olajubu EA. Development of lightweight agents for fault management on computer networks. PhD thesis, Ile-Ife, Obafemi Awolowo University; 2008.
17. Chang EPF, Zhang N. Finding the shortest paths in a large network system. Paper presented at: ACM-GIS 2001. Proceedings of the 9th ACM International Symposium on Advances in Geographic Information Systems; 2001 Nov 09–10; Atlanta, USA. New York: ACM; 2001. P. 160–166.
18. Samarah M, Chan P. Enabling mobile agents communication [document on the Internet]. c2000 [cited 2007 Feb 15]. Available from: https://www.cs.fit.edu/Projects/tech_reports/cs-2000-3.pdf