ORIGINAL RESEARCH

Development of agent-based system for monitoring software resources in a network environment

Akinyokun O. C*[1], Ekuewa J. B.[2], Arekete S. A.[3]

[1] Department of Physical Sciences, Landmark University, Omuaran, Nigeria
[2] Department of Computer Science, Federal Polytechnic Ede, Nigeria
[3] Department of Mathematical Sciences, Redeemer's University, Redemption City, Nigeria

Received: July 21, 2014 **Accepted:** August 13, 2014 **Online Published:** September 11, 2014
DOI: 10.5430/air.v3n3p62 **URL:** http://dx.doi.org/10.5430/air.v3n3p62

Abstract

Mobile agent is becoming an emerging tool for monitoring and managing computer networks. Its usefulness in this regard emanates from its ability to communicate with other agents and devices, and navigate a computer network to collect data and take actions autonomously. In this research, an investigation of the use of an agent-based system to monitor the software tools on the nodes of a computer network is carried out. The proposed framework adopts a multi-agent system approach combining a static server agent with a mobile monitor agent which move around and extract data from each node via the server agent. The system was tested in a computer network environment which is characterized by a Windows NT. The programming and mobility infrastructure is the C#, an object-oriented and multifunctional programming scheme. The performance of the proposed agent-based system and Remote MONitoring (RMON) system are simulated and the results obtained show the cost of service, query time and delay overhead is lower in the agent-based system than that of RMON.

Key Words: Agent-based system, Mobile agent, Network monitoring, Software resources

1 Introduction

The term "agent" originated from the Greek word "agein" which means to drive or to lead. Agent is used to describe something that can produce an effect, for instance, a "drying agent" or a "shipping agent". In Computer Science, an agent denotes a computer system that is situated in some environment and is capable of autonomous actions, for example, a software agent that can search and buy air tickets over the Internet. Though, there can be several agents, in this research, the term "agent" is restricted to a software agent.

Agents have become topical since the 1990s, in particular, in discussions relating to distributed and autonomous decentralized systems.[1] Most of the technologies supporting agent-based systems emanated from distributed Artificial Intelligence research.[2,3] The growing interest generated in the area of agent research is attributed to the significant advantages inherent in such systems, which include their ability to solve problems that may be too large for a centralized single agent, provide enhanced speed and reliability, and tolerate uncertain data and knowledge.

A computer network is a collection of physically separated computers which are connected together primarily to search for, share and exchange computer resources. The process of monitoring software tools on servers and workstations in a network is one of such tedious tasks of the network system administrator. Monitoring and searching for resources on the network often involved physical movement of the network administrator from one computer to another.[4,5] When the human administrators are used for this function, their work may involve monitoring, evaluating and analysis of the various nodes attached to the network with a view to resolving problems and ensuring optimal performance and efficiency. This function can be tiring, stressful and cumbersome, especially in a large network. One significant limitation of this manual approach is that human being cannot monitor events on the network in real time, that is, as the

*Correspondence: Akinyokun O. C; Email: akinwole2003@yahoo.co.uk; Address: Department of Physical Sciences, Landmark University, Omuaran, Nigeria

events occur in nodes distant from where the administrator is currently located. Being human, the network administrator can also be bored and/or confused about which node to monitor next. It is, therefore, apparent that manual network management cannot satisfy the requirements of the modern complex network systems. The limitations of the manual approach necessitate the need to have intelligent software that would autonomously search, detect and monitor network resources on behalf of the network administrator.

Several agents have been described in literature. Certain agents, such as, static agents are stationary and would not need to migrate from place to place.[6] On the other hand, some agents called mobile agents can migrate from one node to another in a network to perform tasks on behalf of the network administrator or a user.[4,7–10] The Distributed Artificial Intelligence (DAI) community includes the intelligent and multi-agent systems with their main focus on agents (stationary) placed at nodes or workstations distributed over the network and cooperating to pursue a common goal. Multi-agent systems consist of a number of autonomous agents that cooperates or compete to achieve some defined goal.[11,12] Mobile agent can be intelligent as well as being part of a multi-agent system, and as such, the DAI community considers mobility an orthogonal or optional property of an agent.[13–15]

The theories, concepts, frameworks, platforms, standards and interoperability of mobile agents and their application in network management, wireless sensor networks, mobile devices, e-commerce, emergency response and other areas have been discussed in details in.[12,16–18] In a network, a software agent can be dispatched from a server to any workstations to monitor software tools available without the system administrator physically moving from one system to another. In this research, an agent application is developed to autonomously monitor and evaluate software tools in a computer network.

2 Development of agent-based system

The mathematical model for agent-based system is presented in Section 2.1. Section 2.2 presents the architecture and design of the agent-based system.

2.1 Mathematical model of agent-based system

A mathematical model[19] using push migration strategy has been adopted in this research. In that model, when an agent migrates to a new location or node, it carries all its code, data and all state information along. The migration process is divided into the following three parts.

1) Mobile Agent (MA) starts off from the server (home) platform, S_h and migrates to the first target node in a given hierarchy.

2) Mobile agent migrate from target node N_k to N_{k+1}, where k=1,2,...,m-1.

3) Mobile agent migrates back to its home platform.

Accordingly, the total network load of MA is segmented into the following three parts:

1) The load of MA denoted by B_h while migrating from S_h to N_1.

2) The load accumulated of MA denoted by B_m while it moves through the target nodes

3) The load of MA denoted by B_f while it moves from the last target node to home node.

The total network load denoted by L is therefore given as:

$$L = B_h + B_m + B_f \tag{1}$$

Let a set of target nodes to be visited defined as:

$$N = \{N_1, N_2, N_3, \cdots N_m\} \tag{2}$$

A mobile agent is composed of the code, data and state information, which are donated by c, d and s respectively. Let the code be composed of n-classes, therefore, the total length of the code in bytes is:

$$B_c = c_1 + c_2 + c_3 + \cdots + c_{m-1} + c_m \tag{3}$$

B_c remains constant throughout its life time. Assume the length of data in bytes of MA at take-off is d_h and at each node visited, accumulates additional data denoted by d_k,k=1,2,3,...,m. Again, assume the length of the state information in bytes is B_s and this is constant throughout the agent life-time. Then, the load B_h of MA from home to the first target node is calculated as:

$$B_h = B_c + d_h + B_s \tag{4}$$

When MA migrates from N_k to N_{k+1} with k=1,2,...,m-1, it has a network load of

$$B_M = B_c + d_h + B_s \tag{5}$$

When the agent migrates to its home, the load is given by:

$$B_f = d_h + B_s \tag{6}$$

The agent-based system comprises a server which connects to a number of workstations. The server is composed of typical computer hardware devices such as main memory, secondary memory, printer, scanner, switches, modems, network ports and so on. There are also some categories of software such as network operating system, frontend software, backend software and utility software. The workstation environment on the other hand comprised of some hardware devices and software systems of, perhaps, lesser capacity than that of the server.

2.2 Architecture of agent-based system

The agent-based system adopts a multi-agent approach: the static agent, otherwise referred to as Server Agent (SA) and the mobile agent referred to as Monitor Agent (MA) together with their underlying software and hardware infrastructure. The architecture of the system is composed of a backend and frontend engine. The backend engine is made up of the server and workstations. The frontend engine provides the framework for launching and migration of the monitor agent. The architecture of the agent-based system for monitoring software tools is conceptualized in Figure 1. The platform for the take-off of the monitor agent at the source and the platform for its landing at the target workstations are their respective operating systems. In monitoring of software tools, the model conceives to main issues. One, a system has to monitor the software tools on workstations in the network, and secondly, a system has to report back to the server where the request is made. A static agent (server agent) is responsible for monitoring at its locality while a mobile agent (monitor agent) is responsible for visiting each node, activating the server agent, getting the information on the software tools and reporting back to the server. The two agents are integrated to make the proposed system as depicted in Figure 2.

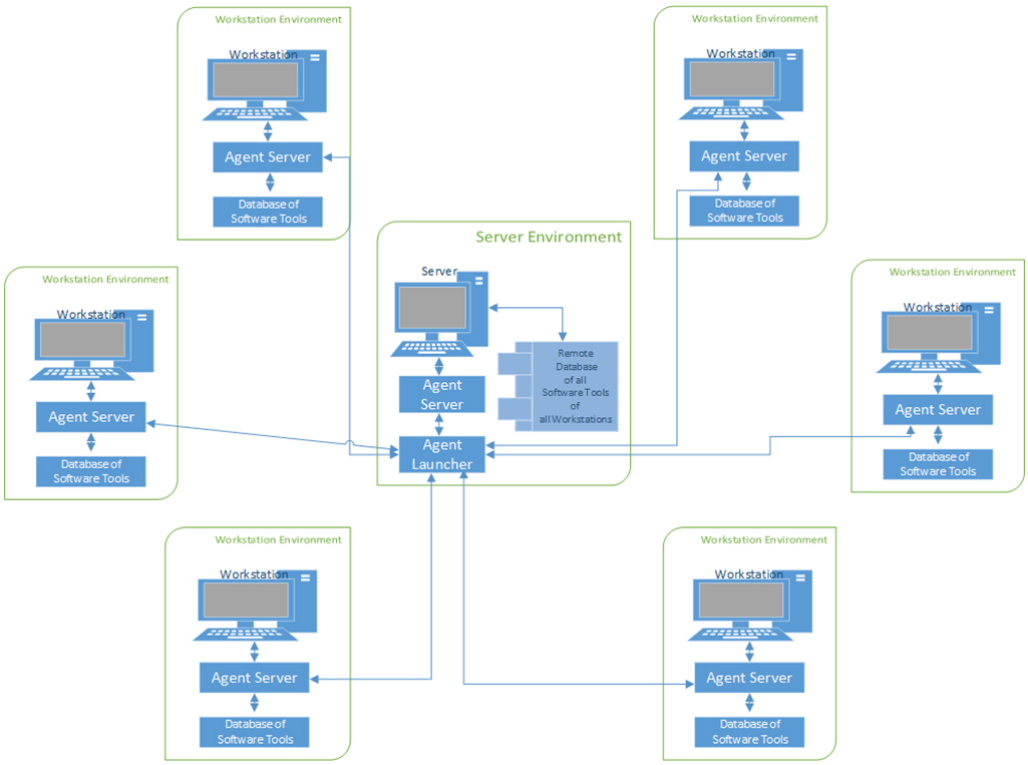

Figure 1: Architecture of the Agent-Based System

2.2.1 Server agent

The server agent is a backend static agent. It executes only on the system where it is installed. It must be installed on both the server and all the workstations in the network. The server agent must be running on all the computers in the network to enable the monitor agent to go into any workstation to do its job and report back to the server which is making such request. The server agent is responsible for performing the following functions:

1) Provide an interface for the user or system administrator to specify requests to the monitor agent.
2) Create monitor agent on behalf of the user or system administrator.
3) Provide avenue for the user to specify travel plan.
4) Launch the monitor agent and migrate it to the next workstation in the itinerary.
5) Keep track of the monitor agent in order to service any special request from other agent servers.
6) Process information results from the workstations visited before presenting it to the user or system administrator in a Graphical User Interface (GUI).
7) Provide computer resources at both the server and at the workstations for the monitor agent.
8) Provide an enabling execution environment for the monitor agent.
9) Provide an environment for the mobile Agent Communication Language (ACL) necessary for the incoming monitor agent to be able to run its code to monitor the software tools.

ISSN 1927-6974 E-ISSN 1927-6982

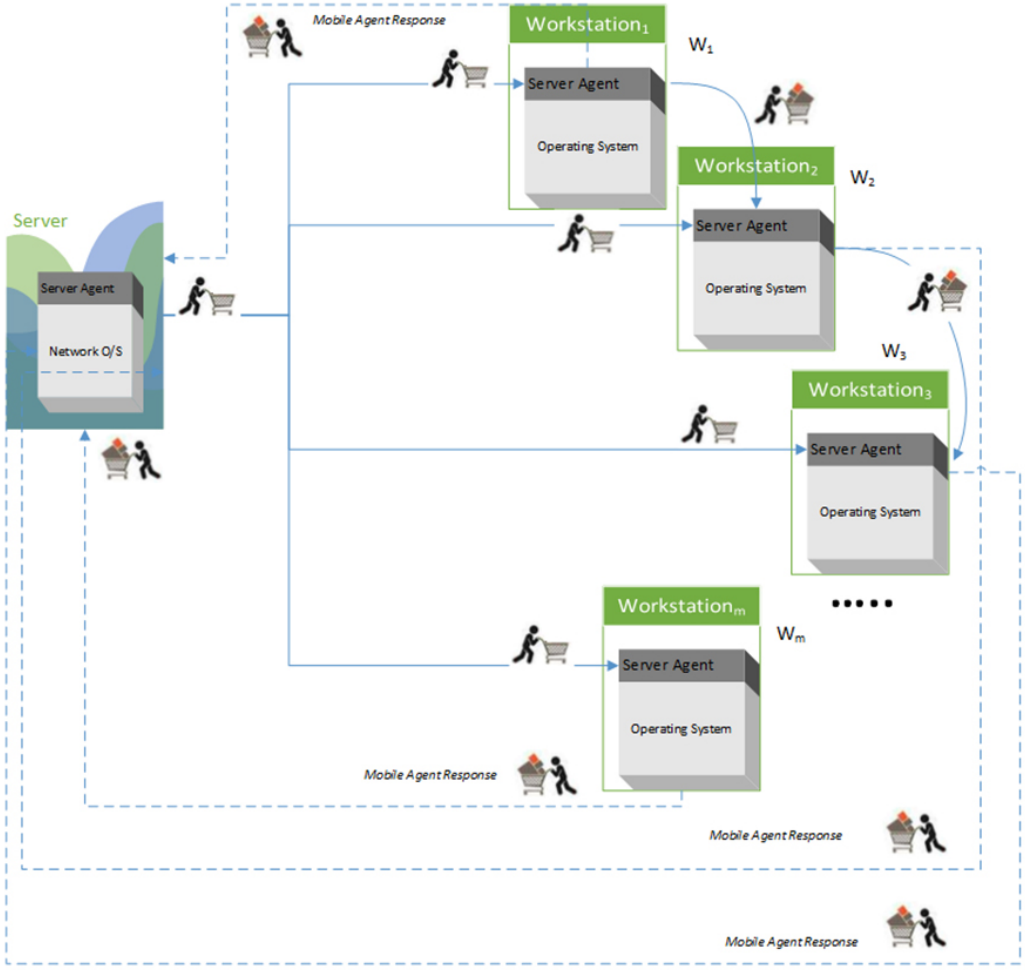

Figure 2: Agent Mobility Model

2.2.2 Monitor agent

The monitor agent is a mobile agent dispatched from the server to other workstations in the network. It goes into the network to identify the software tools on the workstations or servers whose identities are known. At each workstation, the monitor agent interacts with server agent at each node to collect information about the available software tools. The information collected at each node with its identities is placed in the database container of the monitor agent for onward movement to the next workstation. This process is repeated until the last node is visited, at which point, the monitor agent migrates back with all software information in its database container to the server that launched it. At the target workstation, the operating system provides a platform for interaction between the server agent and the monitor agent. The server agent gets into the files of the operating system to collect the information about the software tools on it and places it in the database container of the monitor agent. The database container of the monitor agent is used to update the source server agent from where the collected information is displayed on the screen or printed out for the system administrator.

2.2.3 Mobility facility

Mobility is the core property in a mobile agent concept whereby the agent has the ability to migrate or transport itself from one node to another within the same environment or from node to another node in a different environment autonomously. The model envisages a mobility framework which supports transporting the mobile agent from the server to the workstation, between the workstations and back to the server. Theoretically, migration between the workstation should be unidirectional, that is, if the monitor agent leaves the workstation W_1 for workstation W_2, it should not return to W_1, on the other hand, it should move to the next workstation W_3 in the itinerary or return to the server if the last workstation has been visited. The movement of the mobile agent in the network is depicted in the model in Figure 3.

After the service is started and the server agent is initiated, the mobile agent is launched and initialized. When authentication is successful, the mobile agent is migrated to the workstation where it interacts with the server agent at that node and obtains software tools data. Mobile agent then migrates to the next node and the same process is repeated till

the last node is visited. Mobile agent returns to the server, display result and archive it. Finally, the server agent deactivates the returned monitor agent.

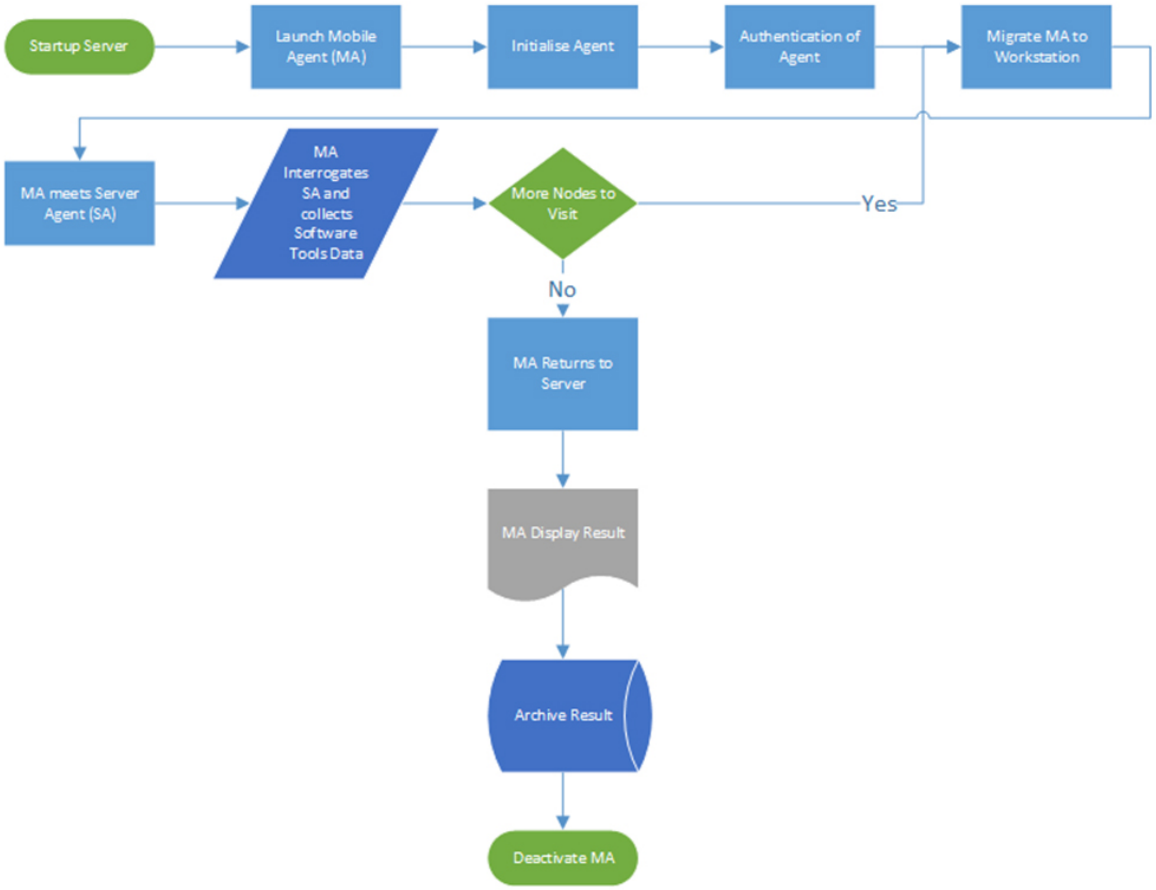

Figure 3: Mobile Agent Migration Flowchart

3 System implementation of agent monitor

The agent-based system requires a network which supports both server and workstations of suitable configurations in a local area network. The Windows NT operating system was used for availability and efficiency. However, the system will operate well in Linux, UNIX and Solaris operating system environment.

The agent-based system adopted the Microsoft Access relational database as the tool for storing system information because it is readily available and cheap to obtain. Moreover, it works seamlessly with other window based tools which were used to implement the system. MS-Access can also exchange data with other relational database systems such as Oracle, SQL Server and Sybase. MS-Access employs a facility called the Microsoft Distributed Transactions Coordinator (MSDTC) which enables clients to make changes to multiple databases at the same time, supports a wide variety of clients that enables users to insert, update, delete and query data stored in databases and works perfectly with Non-Microsoft Access programs, thereby enabling programmers the greater flexibility in creating interfaces that meet their specific network needs.

The frontend provides the interface for the agent to monitor the software tools on the network. The interface software is necessary to assist in Human-Agent-Interaction (HAI). Though in theory, any language can be used to implement mobile agents, a number of languages are known to offer support for agent programming. These include Java, Telescript and Agent TCL. In this research, the C# programming language was used. C# is a simple, modern, general purpose, multi-paradigm and object-oriented programming language which can be used to develop software components suitable for deployment in distributed environments. It is suitable for writing applications for hosted and embedded systems, ranging from the very large programs that use sophisticated operating system, down to the very small ones having dedicated functions. C# can be easily harnessed with Microsoft Access and Windows NT operating platform of the mobile agent as well as with the mobility software. It incorporates features such as menus, forms and command buttons for interactive programming, and these features out-

ISSN 1927-6974 E-ISSN 1927-6982

perform the interactive facilities provided by Java or C++.
For the mobility software needed for mobile agent, several choices are available, which include Java, Telescript, Obliq, AgentTcl, and C#. C# has been adopted as the mobility software in this research. Though C# was not specifically designed for writing mobile agents, but it has most of the necessary capabilities for mobile agent implementation. It has built-in language thread and synchronization functions that are very secured which make programs to run on different platforms in the network. C# programs are compiled to byte-codes (binary instructions) that run on any platform under the Microsoft Common Language Runtime (CLR) which makes C# programs highly portable. It has built-in services which facilitate the mobility of codes such as object remote and serialization. It has security mechanisms built into the Microsoft Common Language Routine (CLR) instruction set to prevent programs from being accessed outside their environment. Sending an object over the wire is therefore a snap with C#.
Access is granted to the mobile agent monitor by typing ADMIN as username and 4190 as password. The login screen shot is presented in Figure 4. The system allows the user three trials in the login procedure, after which it terminates the access process if the login fails. When the login is correct, the welcome screen is activated. After the correct Administrator's name and Password or PIN have been entered, the login option is clicked to move to the next stage which is the welcome screen module shown in Figure 5.

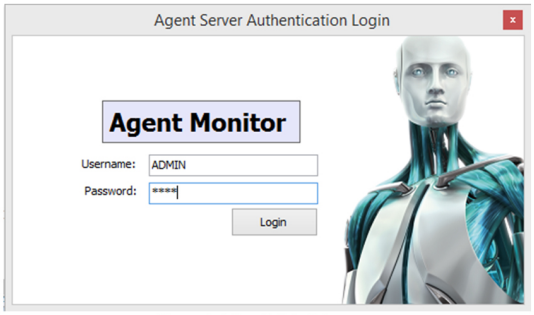

Figure 4: Login Module

Figure 5: Welcome Screen

For the Agent Monitor to identify and connect the computers on the network, all the computers must be configured by IP (Internet Protocol) address to each computer. The IP address can be assigned manually (static) or dynamically by the use of Dynamic Host Configuration Protocol (DHCP). This is used in a wireless Ad hoc network. The nodes on the network request configuration settings using the DHCP such as IP address, a default route and DNS server addresses. Once the client implements these setting, the host is able to communicate on that network. DHCP provides IP addresses automatically so there is no need for manual configuration of IP addresses in the nodes. In this research work, DHCP for dynamic assignment of IP addresses to computers on the network was used because:
1) Dynamic configuration reduces the stress of configuring each and every connected computer on the network.
2) It eliminates the problem of IP conflict that sometimes arises while using static/manual IP address configuration.
3) It reduces the expenses incurred in terms of cables and other accessories needed in wired network.
4) It is portable and can be easily used on mobile equipments.
5) Most of today's computers have built-in DHCP and ad-hoc settings that facilitates for dynamic configuration of IP addresses.
The conceptual diagram for illustrating dynamic configuration of IP address using DHCP is presented in Figure 6.

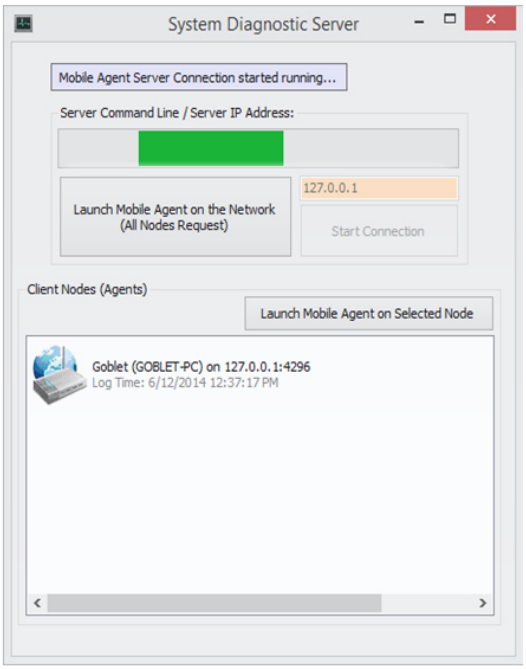

Figure 6: Welcome Screen

The Figure 7 shows the Agent server interface and how it acquires the IP address of the target computer to start listening for an incoming connection from client agents on nodes. DHCP configuration approach was used to make it possible

for the server connection to get client agent IP address immediately they attempt to establish a connection to it. It is mandatory for all monitor agents on the network to know the server IP address. The server does not need to know the monitor agent IP before it connects, but the monitor agent needs to, since it is possible for the server agent to easily retrieve monitor agent IP address and the port from its connection information. The Figure 8 shows how the IP address assigned to the server can be located in the network. Both server agent and monitor agents allow entry of IP address for their communication because, the IP address can change as the computer devices use change. The agent program adopts the IP address entry approach to make the program flexible, dynamic, and easy to implement to prevent hard-coding IP addresses in the program.

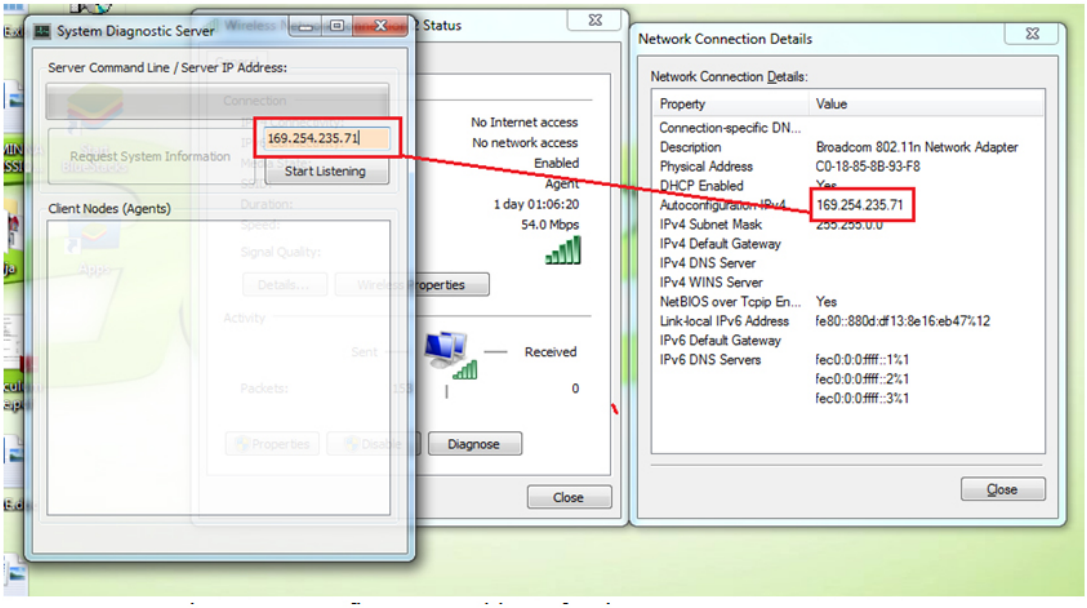

Figure 7: Configure IP Address for the Server Agent

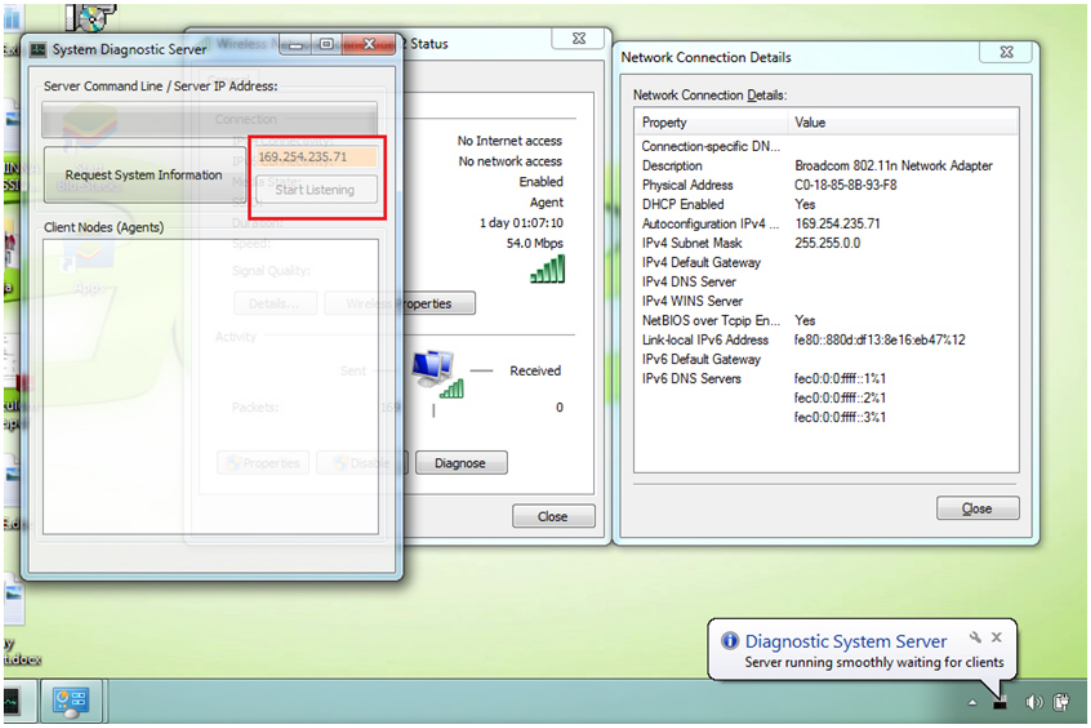

Figure 8: Server Agent Configured and Ready to Communicate with Agents

Figure 9 shows the mobile agent helper program on the target computers. The program helps the agent to gather the required system information, installed programs and task manager running applications for the agent to collect before moving to the next target computer if there is a need for that, depending on the Code/Data Instruction specified for the mobile agent from the server.

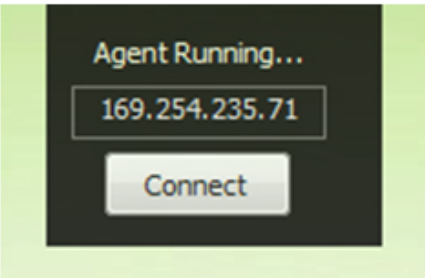

Figure 9: System Information Gathering Agent on Client Computer

The Figure 10 shows a notification popup on the taskbar to notify the user that the server agent has been started listening and waiting for connections from the agent nodes. Figure 11 shows the server agent before listening to node agents. The "Start Listening" button triggers the server agent to accept incoming connections and also respond to them. Before the connection starts, the user on the server-side must preconfigure the listening IP address on which the node agents are to connect. Only agents that are able to connect to the server through the IP address are those the server agent will mount while working on the network to gather information.

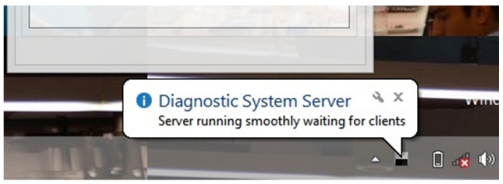

Figure 10: Diagnostic System Server Initialization

The Figure 12 depicts a server agent after setting up connections for the nodes to connect. The server agent has Agents List box that contains all the node agents that are able to connect to the server successfully. With the list box, the agent on the server-side can easily communicate with all nodes or a specific agent on a selected node. The "Request from ALL NODES" button initiates a mobile agent walk-through among all the agents on the network and instructs them to gather and prepare their system information for the server agent. The "Request from Selected Node" button makes it possible for server agent to interact and get system information from a particular node without interfering with the rest. The user on the server can select a node on the Agents List box and then command the agent to connect to the node, gather needed information and return back to the server.

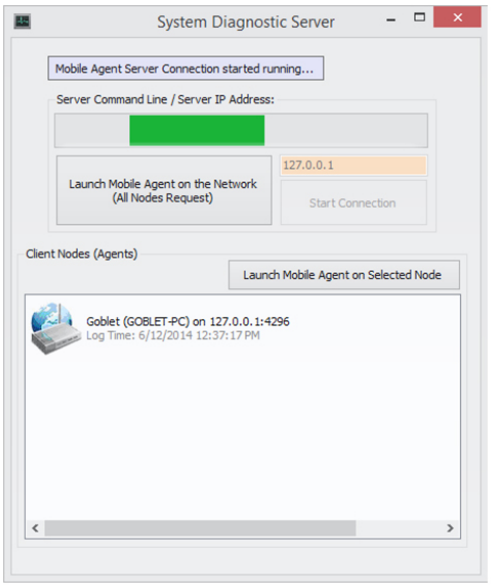

Figure 11: Server Agent before listening to Nodes

Figure 13 displays a notification message dialog box on the server to notify user on the server-side of new incoming system diagnostic reports from the node agents. After clicking "Ok" button, the server collates and processes the incoming data, and it generates a report for each node agent in PDF format. The PDF format is chosen to preserve the processed system information from alteration and to make it portable and organized. PDF file is generated and it contains information about the system such as; Machine Name, Operating System, OS Build Version, Network, Monitor Size, CPU Summary and the Drive information. It also analyzes the running application in each computer memory, their PID (Process Identity), the Maximum Memory to consume (Working Allotted) and the Memory consumed.

The following are the benefits of the system to the administrator:

1) Assessment of system based on software availability on each system from a remote location without visit to the system itself. This will enable the system administrator to know the performance in terms of software of each system.

2) Know the list of software application installed on each node.

3) List of applications that are currently running on each node and the memory space occupied.

4) Get a comprehensive diagnostic information of each node in the network which include: machine name, operating system and its build version, the boot mode, the monitor size, the processor name and its speed, the hard drive capacity and its format (File Allocation Table (FAT) or New Technology File System (NTFS)) which enable the operating system to control how data is stored and retrieved.

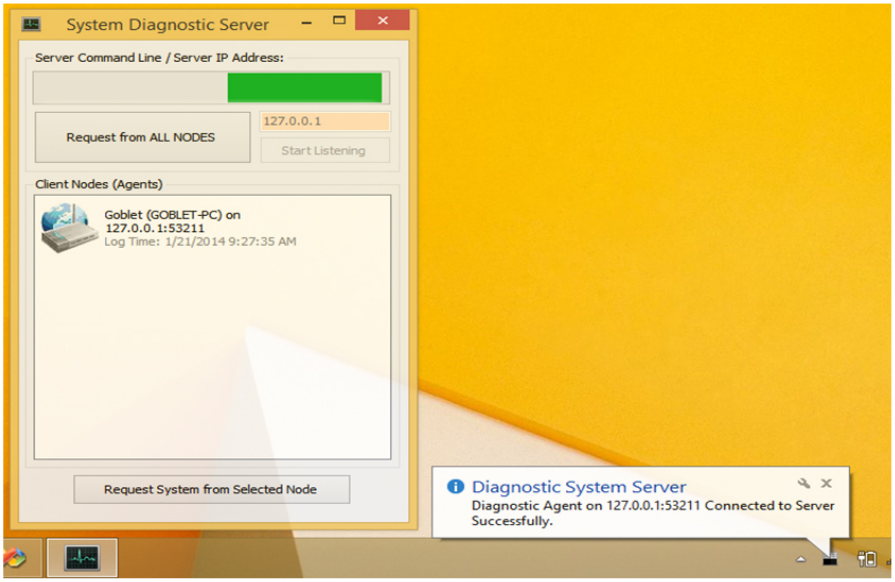

Figure 12: Server Accepts Connection from Node Agent

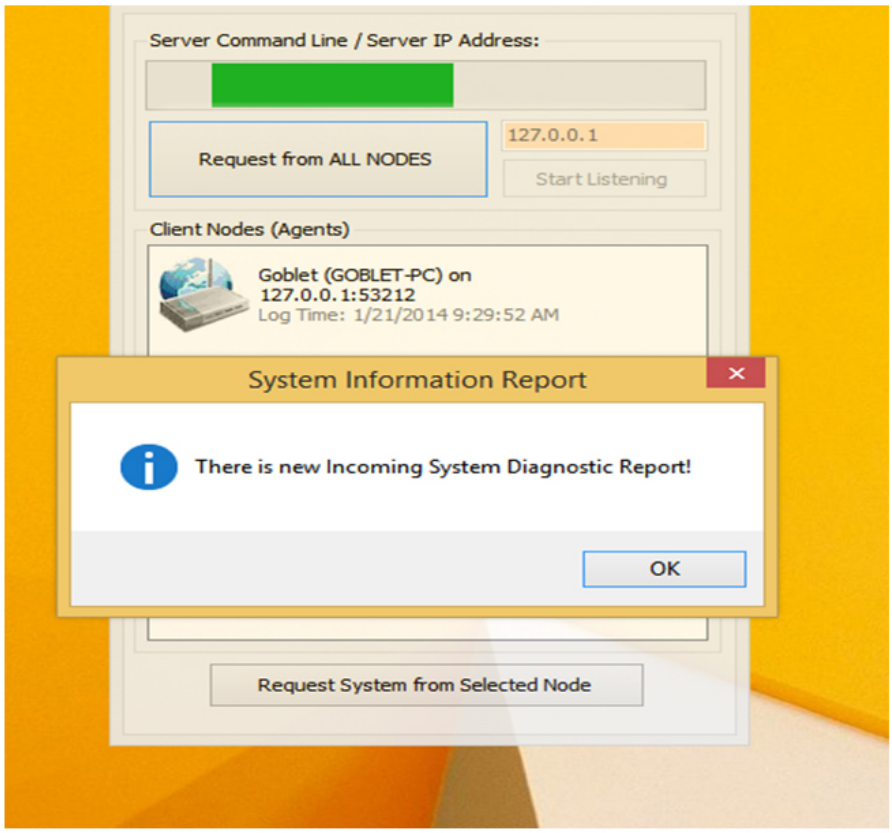

Figure 13: Server Accepts Connection from Node Agent

4 Performane evaluation of mobile agent and rmon

In this section, an attempt is made to justify the performance of the proposed mobile agent with Remote MONitoring (RMON), which is a form of remote procedure calls. In justifying the advantage of the development of agent-based monitor of software tools, three parameters were tested by comparing the gains of mobile agent system with the existing RMON system that uses Remote Procedure Calls

(RPCs). The three parameters that were evaluated are:
1) Cost of service against number of requests per service: the total time it takes in executing series of predefined task such as requests and responses to services from source and destination computers respectively.
2) Query time against number of requests per service: the average number of requests that can be deployed in a service. The optimization of such requests that can be deployed in a service depends on the technique employed in the modeling.
3) Service delay overhead against number of requests per service: the total amount of time it takes a service before it is being attended to during execution.
The following are defined as they apply to the simulation analysis:
1) Service: This is the series of activities involved in execution of a defined task. It involves one or more requests from a source computer to a destination computer and one or more responses in the reverse direction.
2) Request: This is a particular activity within a service, for example, searching a database during information retrieval.
3) Data size: This represents the size (in bytes) of the data to be transmitted on the network.

4.1 Costs of service

In measuring the cost of service, an attempt was made to evaluate what it costs to execute a service given different number of requests per service. In generating a cost model, two resources that are incured in the process of running a service are bandwidth and time. Bandwidth, measured in bits per second (bps), refers to the amount of data transmitted or received per unit of time over the network. Hence, the more the size of data that is transmitted over the network in unit of time, the more the bandwidth requirement. Also the time durations of completing a service would affect the cost of service. Thus, cost of service, C is defined as a function of the total bandwidth required in bits per second (B) and service time (T) in seconds.
A mobile agent executes a service by moving the mobile agent code and all the requests in the service to the destination computer, executes all the requests and then returns to the source with a single response to all the requests. Assuming the size of the mobile agent code is x bytes, then for request and response operations, the total size of the mobile agent code that is transmitted over the network will be $2x$ bytes. The size of the individual requests in the service is assumed to be the same and equal to y bytes. Therefore, for n requests in the service, the total size required is ny bytes. The size of response is assumed to be z bytes. Therefore, the total size of data transmitted for the MA denoted by D_m is given by Equation 7.

$$D_m = 2x + ny + z \tag{7}$$

To calculate the time it takes to transmit the data, it is assumed that the bandwidth of the network is p bps, and that p bits are transmitted per second. Thus, to transmit the total size of data denoted by D_m in Equation (7), it would take time T_m given by Equation 8.

$$T_m = \frac{2x + ny + z}{p} \tag{8}$$

Assuming that the cost of transmitting p bps in 1 second is q units, then for a continuous transmission over a period of time T_m, the cost of transmission, C_m is given by Equation 9.

$$C_m = \frac{q(2x + ny + z)}{q} \tag{9}$$

In RMON, executing a service consists of carrying individual requests in a service and a corresponding response. For requests in a service, the size of the requests is ny, and since n responses would be sent back, the size of the responses will be nz. Thus, the total size of data transmitted over the network for RMON is:

$$D_r = 2x + ny + nz = 2x + n(y + z) \tag{10}$$

Let us assume a network with a bandwidth p bps, the time T_r required to transmit D_r as in 10, then becomes:

$$T_r = \frac{(2x + n(y + z))}{p} \tag{11}$$

Furthermore, assuming that the cost of p bps transmission in 1 second is q units, then for a continuous transmission over a period of time T_r, the cost of transmission, C_r is derived in equation 12.

$$C_r = \frac{q(2x + n(y + z))}{p} \tag{12}$$

From equations 9 and 12, we can see that $C_m < C_r$, that is, mobile agent is more cost effective than RMON.

4.2 Query processing

In this research work, the investigation of how the MA and the RMON schemes execute queries is carried out. For ease of analysis, assume that the resources which the two schemes evaluate from the server of a network has a central storage. Hence, during a service, there are one or more queries at the node. For the RMON scheme, a unique query is carried per unit time for execution. Thus, in a service with n requests, there would be n queries. For the MA scheme, since all the requests are carried out in batches and executed at the node, then repeated requests are not going to be executed twice. It is only the number of unique requests that would be executed. For example, one may have some of

the requests that are the same; hence they would only be executed once. Thus, for MA scheme, we expect that for n requests, there would be m number of queries where m is less than $n(m < n)$.For the RMON architecture, however, if there were n requests per service, then the number of queries would also be n.

4.3 Service delay overhead

Service delay overhead is the overall delay that is associated with a service. The delay associated with the execution of a request for service is classified into three namely: transfer delay, waiting time and service time. It is assumed that no other job is running, therefore, competition for processor time does not occur, and hence, some delays such as interrupts due to other jobs were not present.

The Transfer Delay is the time interval from the generation of the last bit of packet at the information source and when the last bit is recieved at the destination. The main delay components are:

1) Queuing delay.
2) Time at the source interface buffer before the packet is processed for transmission.
3) Processing delay involved as the protocol interpreter is managing the transmission of the packet.
4) Propagation time required to transmit a packet through the network.
5) Waiting time at the buffer associated with the destination station and,
6) Processing delay at the destination station.

The Waiting Time is the time interval between the arrival of a request at the destination and the beginning of its execution. The waiting time in this case is not due to other jobs but due to the execution of the previous request in the service. The Service Time is the time between the start and the end of execution of a particular request in a service. For the MA scheme, the transfer delay is only suffered once during the request operation. However, each of the unique requests would have to be processed one at a time and thus each request suffers some waiting and service time.

Given n requests per service with m unique requests generated, the transfer delay for the request operation is denoted by T. For the first request, there is no waiting time since the request is the first one, hence, waiting time for the first request denoted by w_1 is zero. For the second request, the waiting time is equivalent to the service time of the first request denoted by s_1. Similarly, for the third request, the waiting time is equivalent to the service time of both the first and second requests that $s_1 + s_2$. Assuming that the service time for each request is the same, therefore, the sum of the waiting times follows an arithmetic progression with a common difference s1 or s_2. The total waiting time denoted by W is then found to be:

$$W = (m(2w_1 + s_1(m-1)))/2 \tag{13}$$

It should be recalled that w1 = 0, the expression becomes

$$W = (m(s_1(m-1)))/2 \tag{14}$$

Since $s_1 = s_2 = \cdots = s_m$, then the total service time denoted by P is given by:

$$P = ms_1 \tag{15}$$

Also, during response operation, the MA also suffers a transfer delay denoted by R. Therefore, the total service delay overhead for MA scheme is: $MA(delay) = P + W + S$
Since $P = T + R$

$$MA(delay) = T + W + S + R \tag{16}$$

If we assume equal delay is suffered during request and response operation then $T = R$

$$P = 2T = 2R \tag{17}$$

Therefore, combining equations 14, 15 and 17, MA(delay) is give by:

$$MA(delay) = 2T + (m(s_i(m\breve{}1)))/2 + ms_i, (i = 1, 2, 3, \cdots, m) = 2T + (m^2 s_i + ms_i)/2, (i = 1, 2, 3, \cdots, m) \tag{18}$$

For the RMON scheme, requests are transferred and executed one at a time hence, creating an overhead in transmission delay. However, total waiting time here is zero since a request completes execution before the next is transferred. Therefore, the following applies for the RMON scheme. The total transfer delay denoted by P is given by:

$$P = n(T + R) \tag{19}$$

Since each request consists of a request-response pair where T = R, then,

$$P = 2nT = 2nR \tag{20}$$

The total waiting Time (W) is zero. Also since $s_1 = s_2 =$

$\cdots + sn$, then the total service time P is given by:

$$P = ns_i, (i = 1, 2, 3 \cdots, n) \quad (21)$$

Therefore, the total service delay overhead for the RMON scheme is given by:

$$RMON(delay) = 2nT + ns_i, (i = 1, 2, 3 \cdots, n) \quad (22)$$

4.4 Cost of service versus number of request per service

The mathematical basis for the number of requests per service developed in Equations 9 and 12 that represent the cost of services for the mobile agent monitor and existing RMON schemes respectively is being applied here. The variation of cost with number of requests per service is being simulated. The size of agent codes is assumed to be 5 bytes, size of request y and response z is fixed at 1 byte each. The bandwidth size p is assumed to be 5Kbps and cost of transmitting at 5Kbps for 1sec q is assumed 1 unit. The result of the simulation is as shown in Figure 7. It is clear that when the number of requests increased, the advantage of MA over RMON is more pronounced.

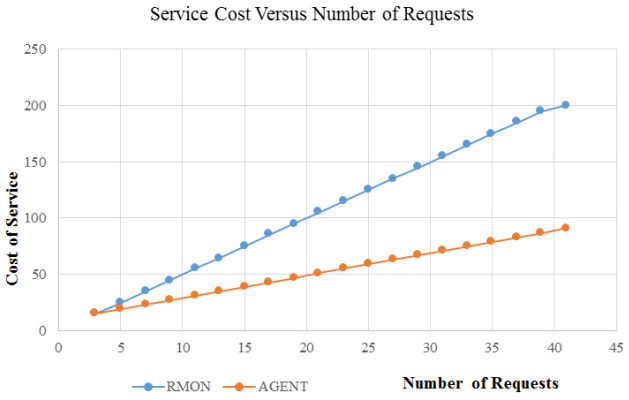

Figure 14: Service Cost of MA against RMON

4.5 Query time versus number of requests per service

From the analysis done in the query optimization, a query time is assigned to each of the unique queries involved in executing a service in the two schemes. For simplicity, assume a uniform query time of 1 second for each of the requests. The result obtained is as shown in Figure 15. The mobile agent scheme optimizes querying time as the number of requests per service increases because it is able to eliminate repeated requests and thereby reduce the number of queries to be executed at remote locations.

4.6 Service delay overhead versus number of request per service

The service delay overhead against the number of request per service for the two schemes was measured. Equations 19 and 22 were adopted for mobile agent and RMON respectively. In the simulation, it was assumed that time delay =10secs, service time = 2secs. The result of the simulation depicted in Figure 16 shows that the MA scheme generates a lower service delay overhead than the RMON scheme. At 18 out of the 20 samples simulation runs, the mobile agent perform better resulting in 90% efficiency.

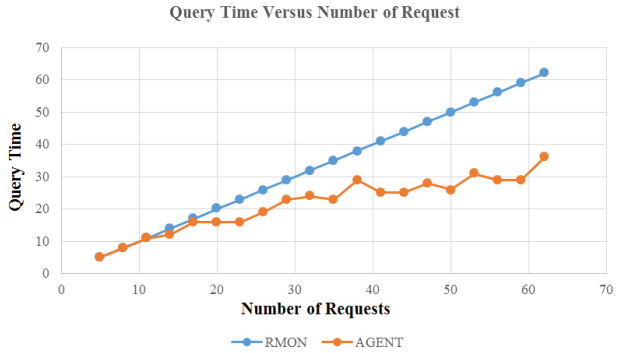

Figure 15: Query Time Versus Number of Requests

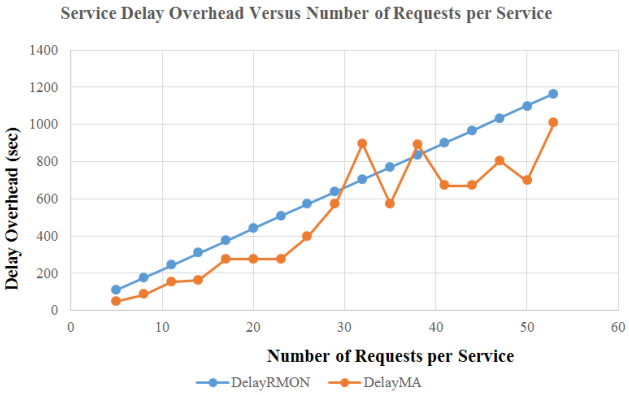

Figure 16: Service Delay Versus Number of Requests per Service

5 Conclusions

In this paper, an agent-based system has been developed to monitor software tools available on the nodes of a computer network. The agent system employs the multi-agent paradigm in which agents interact and cooperate with each other to achieve a common goal. The static server agent seats on each node and collects the data on software tools. The mobile monitor agent then moves into the workstation and interacts with the server agent, receives the data and moves to the next node or the server where the reports are displayed and achieved. The advantage of the agent-based system is that it can monitor each node and identify the software tools installed on them on real-time basis. Information

obtained can be used by the system administrator to take critical decisions. In a large network environment, the work of monitoring software tools can be a tasking one. The proposed system would assist the system administrator to function more efficiently. In future this work can be expended to involve configuration management, fault detection and security.

An attempt is made to justify the performance of the proposed mobile agent with Remote MONitoring (RMON), which is a form of remote procedure calls. In justifying the advantage of the development of agent-based monitor of software tools, three parameters were tested by comparing the gains of mobile agent system with the existing Remote Monitoring (RMON) system that uses Remote Procedure Calls (RPCs). The three parameters that were evaluated are:

1) Cost of service against number of requests per service of the agent system were shown to be lower than that of RMON.

2) Query time against number of requests per service of the agent system were shown to be lower than that of RMON.

3) Service delay overhead against number of requests per service of the agent system were shown to be lower than that of RMON.

References

[1] Arai T and Ota J. "Motion Planning of Multiple Mobile Robots using Virtual Impedance". Journal of Robotics Mechatronics. 1966; 8(1): 67-74.

[2] Guttman, R.H and Maes P. "Agent-Mediated Integrative Negotiation for Retail Electronic Commerce", Proceedings of AMET'98. 1998: 77-90. Available from: http://ecommerce.media.mit.edu/

[3] Feridum, M and Krause, J. "A Framework for Distributed Management with Mobile Components", Computer Network. 2001; 35: 25-38. http://dx.doi.org/10.1016/S1389-1286(00)00147-X

[4] Imianvan A.A. "Development of Mobile Agent for Evaluating the Use of Bandwidth in a Computer Network", PhD Thesis in the Department of Computer Science, Federal University of Technology, Akure, Nigeria. 2009.

[5] Arekete, S.A. "Development of a Mobile Agent for Monitoring and Evaluation of Activities of Users in a Network Environment". PhD Thesis, Department of Computer Science, Federal University of Technology, Akure. 2013.

[6] Akinyokun, O. C. "Catching and Using the Virus". The journal of the Institute of the Management of Information Systems (IMIS), London, United Kingdom. 1997; 7(6): 12-17.

[7] Akinyokun O. C. and Imianvan A. A. "Experimental Study of Bandwidth Management in a Computer Network Environment". Proceedings of Allied Academies International Conference, Orland, USA. 2010. PMid:20543357

[8] Imianvan A. A, Akinyokun O. C, Obasohan E. E. and Obi J. C. "Prototype of an Intelligent Trade Agent". World Journal of Applied Science and Technology. 2011; 3(2): 40-47.

[9] Arekete S. A, Akinyokun O.C, Olabode O. and Alese B.K. "Design of a Mobile Agent for Monitoring Users Activities". Computer Engineering and Intelligent Systems. 2013; 4(2): 33-48. Available from: www.iiste.org

[10] Arekete S.A. and Akinyokun O.C. "Implementation Techniques of Mobile Agent for Monitoring Activities of Users". WebPub. 2013; 1(3): 38-54, Available from: http://www.researchwebpub.org/wjsr

[11] Wooldridge M. "An Introduction to Multi-Agent Systems", John Wiley & Sons Limited England. 2002..

[12] Manvi, S. S and Venkataram, P. "Application of Agent Technology in Communication: a Review", Computer Communication Journal, Elsevier. 2004; 27: 1493-1508. http://dx.doi.org/10.1016/j.comcom.2004.05.011

[13] Wooldridge M and Jennings N.R. "Intelligent Agent: Theory and Practice". In Knowledge Engineering Review. 1995; 10(2): 115-152. http://dx.doi.org/10.1017/S0269888900008122

[14] Lange, D and Oshima, M. "Seven Good Reasons for Mobile Agents", Communications of the ACM. 1999; 42(3). http://dx.doi.org/10.1145/295685.298136

[15] Tveit. "A Survey of Agent-Oriented Software Engineering". 2001. Available from: http//www.abiody.com/jfpa/publications/AgentOrientedSoftwareEngineering

[16] Silva L.M., Soares G. Martins P, Batista V and Santos V. "Comparing the Performance of Mobile Agent Systems: a Study of Benchmarking", Computer Communications. 2000; 23(8):769-778. http://dx.doi.org/10.1016/S0140-3664(99)00237-6

[17] Cucurull J, Martí R., Navarro-Arribas G, Robles S, Overeinder B, Borrell J. "Agent Mobility Architecture Based on IEEE-FIPA Standards", Computer Communications. 2009; 32(4): 712-729. http://dx.doi.org/10.1016/j.comcom.2008.11.038

[18] Manzoor U. and Nefti, S. "QUIET: A Methodology for Autonomous Software Deployment using Mobile Agents", Journal of Network and Computer Applications. 2010; 33(6): 696-706. http://dx.doi.org/10.1016/j.jnca.2010.03.015

[19] El-Gamal Y Khalid E and Magdy S. "A Comparative Performance Evaluation Model of Mobile Agent Versus Remote Method Invocation for Information Retrieval". World Academy of Science, Engineering and Technology. 2007: 286-291.