

Design of a Mobile Agent for Monitoring Activities of Users

[*]S. A. Arekete, [+]O.C. Akinyokun [+]O.Olabode [+]B.K. Alese
[*]Mathematical Sciences Department, Redeemer's University
Redemption City, Mowe, Ogun State Nigeria
P.O. 7914, Ikeja, Lagos, Nigeria
Tel: +234-803-742-9795 E-mail: Samson_Arekete@yahoo.com

[+]Computer Science Department
Federal University of Technology
P.M.B. 704, Akure, Nigeria

Abstract

Monitoring is an aspect of network management aimed at ensuring optimal performance of the network and that the users play by the rules. This paper presents the design of a mobile agent for monitoring the activities of users in a network. Users' activities can be localized on their personal workstation or extended to the enterprise network and the Internet, in which case it can impact on the subscribed bandwidth, which is a shared resource of the corporate entity that they represent. All users hope to work in an environment of unlimited resources, including disk space, RAM and bandwidth. However, though the cost of these computing resources have reduced significantly owning to advances in microelectronic technology, they are still far from being free and inexhaustible. In this research, we design a mobile agent tool that can monitor users' activities in a network environment with a view to enhancing the effective utilisation of system resources, and in particular, the physical memory. The aim is to enable optimal resource utilisation in the network environment. In this paper, we define a mathematical formulation of user activities, load overhead of mobile agents and itinerary partition to avoid the over-bloating problem. The architecture of the mobile agent is explained.

Keywords: Mobile Agent, System resources, Computer network, Code Mobility, Monitoring

1. Introduction

Mobile agents are computing software entities that can start operation on a network node, suspend action there and migrate to another node, and resume operations from where they left off on the previous node. A mobile agent can follow an itinerary and possibly return to the source node having completed a circle. A mobile agent as an execution unit is able to migrate autonomously to another host and resume execution there, continuing from where it left off. In concept, a mobile agent is able to move its whole virtual machine from host to host; it owns its code but not the resources for execution which is owned by the host. While on a particular host, the mobile agent can carry out a number of tasks, and afterwards, it can migrate to another host to continue operation from where it stopped action on the previous host.

As an executing program, a mobile agent is made up of code, data and execution state and is embedded with some intelligence and the ability to autonomously migrate across the network. As it moves, the mobile

agent can represent its owner in various tasks and satisfy its requests. It can communicate with the host as well as other agents on the host; it is able to sense its environment and carry out a set of activities to attain its mission and achieve its goals. The mobile agent paradigm has become topical since the mid-1990s having found applications in several areas. Mobile agent paradigm derives from two basic disciplines – artificial intelligence from where the concept of an agent emanates and distributed systems which defines the concept of code mobility.

The characteristic features of agent paradigm (Al-Kasassbeh and Adda 2007) include:

a. autonomy: an agent operates without intervention;
b. social ability: agents interact with each other through agent communication language;
c. goal driven: agents exhibit goal-directed behaviour; and
d. reactivity: agents perceive and respond to their environment.

The rest of the paper is organised as follows: section 2 examines some related works, in section 3 we present a mathematical model of the users' activities of interest in the research. In section 4, we discuss the architecture of the mobile agent. Section 5 covers discussions and conclusions.

2. Related Works

Mobile The agent may follow a set of rules predefined by the owner and then applies them. The key aspects of agents are their autonomy and abilities to reason and act in their environment, as well as to interact and communicate with other agents to solve complex problems (Jain, Aparico and Singh 1999), (Papazoglou 2001). According to (Shih 2001), a software agent must possess any two of these three behaviours: autonomy, cooperation, and learning. Autonomy means that the agent can act without the direct intervention of humans or other agents and that it has control over its own actions and internal state. The agent must communicate with the user or other agents to receive instructions and provide results. Cooperation indicates a social ability of an agent to interact with other agents or human via some communication language in order to fulfil certain mission or goal. Mobile agent paradigm holds the promise of enhancing the throughput of every computing element in the network and creating a ubiquitous powerful computing environment (Chen and Hu, 2002).

Mobile agents have been vastly considered in many application areas such as e-commerce, network monitoring and management, distributed information retrieval, telecommunications, remote device control and configuration, Internet etc. (Pullum 2001), (Jin, Ahn and Lee 2009), (Boutaba and Xiao 2002).

In this research, we focus on one vital area of network management, that of monitoring users' activities on the network. In today's complicated and diverse networks on which the survival of corporate businesses are anchored, efficient management of corporate networks have become non-negotiable and imperative. We

face the great challenge of effectively managing these networks in a society that has become heavily dependent on accurate information and its timely exchange and availability. Not just that the network must be up and running all the time, there is also a growing concern for security issues, denial of services and unauthorized access. Human administrators of network systems have been used. Their work involves monitoring, evaluation 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 and cumbersome, especially in a large network. There are usually delays in monitoring events on the network and these events are usually not monitored in real time, that is, as they occur. Being human, the network administrator can be, slow, bored and may be 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.

Different computer-based approaches to effective management of computer networks have been proposed in literature. To this end, various types of network management paradigms have emerged (Martin-Flatin and Znaty 1997). The most popular approach to manage networks comes from the Internet Engineering Task Force (IETF), based on the Simple Network Management Protocol (SNMP). Closely related in structure is the approach based on the Common Management Information Protocol (CMIP) proposed by the International Organization for Standards (ISO) for application within open systems interconnection (OSI) networks. Both approaches assume the presence of management stations (MSs) that interact with management agents running on network nodes. The agents in these protocols are computational entities responsible for collecting and storing management information local to the node and responding to requests for this information from the MS via a management protocol that specifies the packet format for a set of basic operations. The MS interacts with the agents using client/server architecture wherein the functionality of both clients (managers) and distributed servers (agents) is rigidly defined at design time. Within these protocols, physical resources are represented by managed objects and collections of managed objects are grouped into tree-structured management information base (MIB). Both SNMP and CMIP are typically associated with massive transfers of management data, which cause considerable strain on network throughput and processing bottlenecks at the manager host. This centralisation in network management seriously limits its scalability (it is not efficient when the network nodes grow significantly), leading to poor performance and inability to cope with the dimension of the network. As a result, several hierarchical/distributed management frameworks have been proposed by researchers and standardisation bodies. However, these models are typically identified by static management components that cannot adapt to the evolving nature of today's networks, with rapidly changing traffic patterns and topology structures.

In view of these limitations, new distributed technologies have been proposed, which are based on mobile code or distributed object paradigms. Typical research efforts in that direction is (Carzaniga, Picco and Vigna 1997) which reviewed mobile code paradigms used by distributed applications. Mobile agents exhibit certain characteristics that allows for effective deployment of services and applications in a more dynamic, flexible, and customizable manner in a distributed systems. These features make them more

desirable for building mobile applications over the traditional client-server paradigm (Pullum 2001). These features include:

- Mobile agents naturally operate in heterogeneous environments characterised by wide-area and diverse networks where either the reliability or the security assumptions of the computers and the network connections respectively are not a factor (Qu, Shen and Defago 2005).
- Mobile agent acts autonomously on its own or on behalf of the user with the ability to precisely initiate its migration by itself and execute its owner's request and move independently from one host to another in order to resume execution (Park, et al. 2002).
- Mobile agents exhibit a multi-hop ability, that is, they are able to travel with their code, data, and execution state more than once by resuming their execution in another server in the network after completing their tasks in the first server visited. In contrast, mobile code is transferred only once in other mobile paradigm (Park, 2004).
- Mobile agents heavily rely on the underlying protocol for communications by way of interactions and message exchanges in order to successfully carry out and execute certain task in the in the agent system (Qu, Shen and Defago 2005). During migration, mobile agents can communicate synchronously or asynchronously with other agents or the host systems. Others are through network-based mechanism and local inter-process communication mechanism.

Some of the benefits provided by mobile agents that make them preferable for the creation of distributed systems and particularly mobile agent applications than other paradigms include the following:

- They conserve bandwidth especially in applications where there is large amount of data to process on different hosts; code move to the data site rather than the data to the code (Braun and Rossak 2005).
- They can exhibits asynchronous and autonomous interaction since they can be delegated to carry out a certain task and with the intelligence embedded on them, can decide on its own whom to interact with at execution time (Park, 2004).
- They offer extended flexibility in disconnected data operations especially in environments with intermittent connection such as the Internet (Gray, et al. 2000). Unlike client-server application, it can execute the requested operation on a specified host and wait until connection is available before they migrate to the next host.
- They can improve network latency with better response time than client-server applications since the intermediate results are not sent back and forth between client and server, which may have resulted in delays caused by network congestion (Gray, et al. 2000). In this case, only the final result of the computation will be transmitted to the server.
- They are robust and fault tolerant because of their ability to dynamically respond to unfavourable conditions and events in a distributed system (Kotz and Gray 1999).
- They provide support for heterogeneous environments through their mobility frameworks, which separated the executing host and its operating system. Mobile agents are generally independent of the computer-layer and transport-layer and dependent only on their execution environment (Lange and

Oshima 1999).

- Mobile agents have better scalability especially with respect to their being flexible in dynamic deployment (Gray et al. 2000). In client-server development, modification of the client may require redeployment of client installation. In mobile agent systems, on the other hand, the modified agent can be injected once and installs itself wherever it is required to install.
- Mobile agents have been proposed for load balancing routing for LEO satellite networks (Rao and Wang, 2010).

In (Imianvan 2008) and (Akinyokun and Imianvan 2009) a proposal is made for an experimental study of bandwidth management in a computer network environment. The studies were motivated by the wide applications of mobile agents in various fields of human interest including e-commerce, distributed information retrieval and network management. Proper network management is viewed to require monitoring and controlling the resources of the network and assessment and evaluation of network resources are thus crucial part of network management activities. The two-fold objectives of the study were: to provide an intelligent system that can police the use of bandwidth in a computer network environment, and to generate desirable statistics for policy formulation and decision making. Their research methodology involves a design of a prototype expert system for the management of bandwidth use in a computer network was developed and tested; the system is characterized by UML, Z-notation, Petri Nets, Telescript mobility and object-oriented programming. The mobile agent system was made up of server machine which connect a number of workstation; the mobile agent could be launched from the server machine to survey the state of the resources in any of the workstations in the network. Telescript commands were used to drive the mobility of the mobile agents. A mathematical modelling of bandwidth use was formulated and a case study of local area networks of the University of Benin was made, data were collected and analysed. The implementation was based on an environment which is characterized by Microsoft Windows NT Operating System as the platform, SQL Server 2000 SP3 as the backend engine and Visual Basic Application Language as the front end engine.

That research provided a mobile agent system that has practical capability to assess and evaluate the use of bandwidth in a computer network environment. The result obtained from the experimental study provided information for ranking workstations by bandwidth consumption, statistical insight into user's data processing activities and their corresponding consumption of bandwidth and monitoring and control of bandwidth by user with a view to effecting redistribution, prioritization and sharing. The usage of bandwidth released by the ISP to clients were tested and measured on hourly basis to determine the efficiency. The research was however limited in the following respects:

a. the experimental study was tested in local area networks environment where Internet connectivity was unavailable;
b. the study was limited to bandwidth, which is one of the many resources of the network that could be managed; and finally

c. the mobile agent was developed using visual basic which is not particularly well suited for mobile agent technology.

In this research, we seek to follows-up on the work of (Imianvan, 2008) and to extend the work in other areas of monitoring and evaluation of user's activities in network environment. A robust design of a mobile agent system for monitoring and evaluation of user's activities in a network environment was carried out using mathematical, graphical and formal modelling tools. A mathematical model was formulated for transmitting the code, data and information status of a mobile agent system for monitoring and evaluation of user's activities in a network environment; Z specification language was be adopted in modelling the mobile agents. Unified Modelling Language (UML) was employed to handle interaction of the mobile agent system. Implementation is proposed in Java, a programming environment that is mobile agent compliant and reputable for its portability, mobility and object-orientation.

The functional goal of this research is to apply the mobile agent technology in monitoring and evaluation of the activities of users' in a network environment. The specific objective are: to design a model of a mobile agent for monitoring and evaluation of activities of users in a computer network environment, and develop a mobile agent capable of self-cloning and autonomous mobility using JADE Mobile Agent Platform and Java technology.

3. Mathematical Model

Activities of a user could impact on the resources of the local node and the network. Of particular interest in this research is physical memory usage of the system. Users' activities reflect in terms of processes initiated by the user (denoted as Q) which may run concurrently. When a computer system is started, a number of processes are also initiated by the operating system. These fall into two categories: the compulsory processes (R) and the optional processes (S). In the face of scarce node resources, the optional processes can be terminated or killed to improve the performance of the system.

Let P denote the total processes running on a node defined by:

$$P = \sum_{n=1}^{3} p_n \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$\text{Let} \qquad Q = \sum_{i=1}^{d} p_i\,, \qquad R = \sum_{j=1}^{d} p_j\,, \qquad S = \sum_{k=1}^{d} p_k \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

where *d* is finite.

Hence,

$$P = Q + R + S \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Each process utilizes the resources of the computer system to varying degrees depending on its length

or size in (bytes). We assumed that:
Total physical memory is denoted as M_t
Memory consumed by process p_i is m_i
Total memory consumed by all processes running on a node (M_p) is given as:

$$M_p = \sum_{i=1}^{n} m_i \quad \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Percentage memory consumed by all processes:

$$\%MC = \frac{M_p}{M_t} \times 100\% \quad \ldots\ldots\ldots\ldots\ldots\ldots (5)$$

We classified the health status of a node as a function of the available physical memory as follows:

$$\%M_{av} = 100 - \%MC = \begin{cases} \geq 80\% & - Perfectly\ Healthy \\ 60 - 79\% & - Very\ Healthy \\ 50 - 59\% & - Healthy \\ 40 - 49\% & - Slightly\ Healthy \\ 30 - 39\% & - Slight\ Unhealthy \\ 20 - 29\% & - Unhealthy \\ \leq 19\% & - Very\ unhealthy \end{cases} \quad \ldots\ldots\ldots (6)$$

The mobile agent A visited each node and extracted data on the configuration and application processes. This data was carried along to each of the nodes the mobile agent visited on its round trip. Along with the code and its information status, this data constituted a load on the network as the mobile agent moved along. A mathematical model for transmitting the code, data and information status of mobile agent from source to destination platform was formulated, adapting the push-all-to-next strategy in line with (El-Gamal, El-Gazzar and Saeb 2007) and (Braun and Rossak, 2005), whereby, when an agent migrates to a new location it carries all its code, data and all information status along.

We assumed an ordered set of target nodes to be visited by the mobile agent, defined as:

$$T = \{t_1, \ \ldots, \ t_m\} \quad \ldots\ldots\ldots\ldots\ldots\ldots (7)$$

Mobile agent is made up of three components: codes *c*, data *d*, and information state, *s*.
If the code is composed of n classes, the total length of the code (in bytes) is:

$$B_c = \sum_{k=1}^{n} C_k = c_1 + c_2 + \cdots + c_{n-1} + c_n \quad \ldots\ldots\ldots\ldots\ldots\ldots (8)$$

where B_c remained constant throughout its life time.
We assumed that the length of data (bytes) of A at take off is r_h and at each node visited, it accumulated additional data r_k, $k = 1, \ldots, m$.
We further assumed that the length of the information state (bytes) is B_s, and this is constant throughout the agent lifecycle.

Then, the load B_h of A from home to the first target node was calculated as:

$$B_h = B_c + r_h + B_s \quad \ldots\ldots\ldots\ldots\ldots\ldots (9)$$

A migration from T_k to T_{k+1}, with $k = 1, \ldots, m-1$ has network load of:

$$B_m = B_c + r_h + \sum_{k=1}^{m-1} r_k + B_s \quad \ldots\ldots\ldots\ldots\ldots\ldots (10)$$

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

$$B_f = r_h + \sum_{k=1}^{m} r_k + B_s \quad \ldots\ldots\ldots\ldots\ldots\ldots (11)$$

An itinerary partitioning process is modelled in line with (Verma, et al. 2008). We consider *m* nodes to be managed. The initial itinerary I_{init} for the MA is represented as:

$$I_{init} = \{S_h, t_1, t_2, \ldots, t_m\} \quad \ldots\ldots\ldots\ldots\ldots\ldots (12)$$

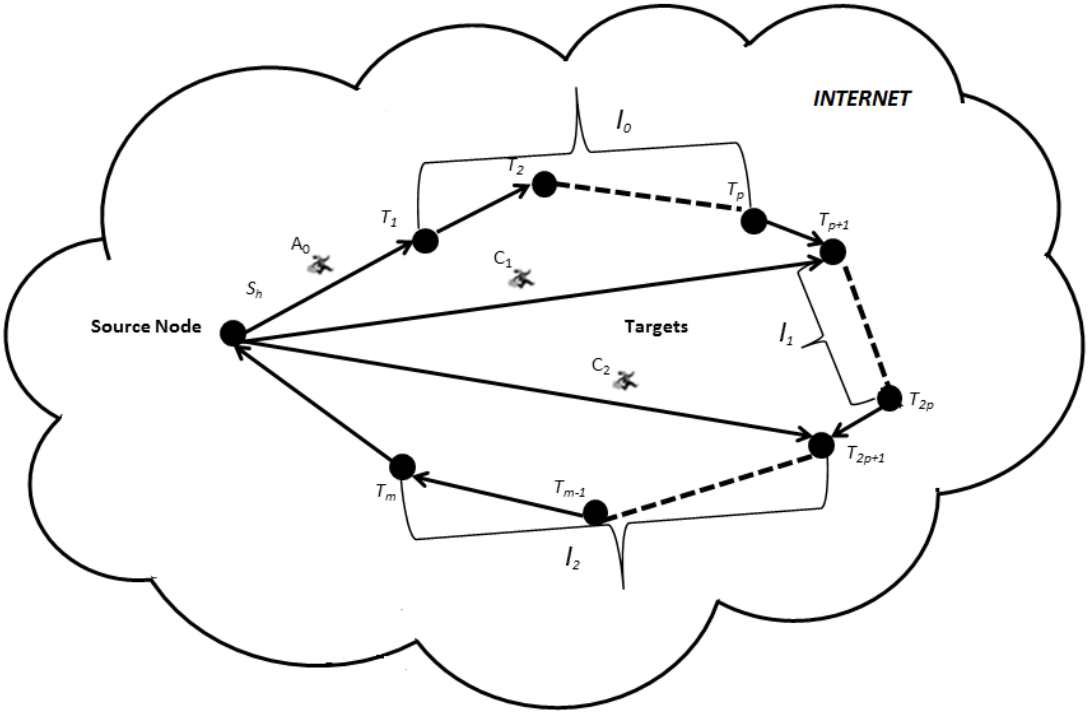

Figure 1 Multi-agent system with an agent visiting a sub-itinerary

where S_h is the home node and $t_i (i \geq 1)$ is the *i*th monitored node in the network. When the number of nodes to be monitored (*m*) are very large, the response time for a single mobile agent visiting all the nodes sequentially become very high, resulting in bloated state problem (Verma, et al. 2008) - when the accumulated data from node to node would have been too much. We therefore consider using multiple agents by making clone(s) of the original MA. We define a partition factor *p* to control the

itinerary partition. The partition factor $p(1 \leq p \leq m)$ is the maximum number of nodes that an MA can visit. We define sub-itinerary, I_j:

$$I_j = {}^{m}/_{p} \quad \ldots\ldots\ldots\ldots\ldots\ldots (13)$$

From eq. (13), the initial itinerary is therefore divided into m/p sub-itineraries. When p=1, it has m sub-itineraries, that is, each MA will visit only one node. On the contrary, when p=m, the initial itinerary has no partition, meaning that a single MA travels through the entire itinerary. In general, when MA arrives at node T_1, it examines the itinerary. If m>p, the itinerary is partitioned into two independent sub-itineraries, I_1 and I_2.

The sub-itineraries are:

$$I_1 = \{t_1, t_2, \ldots, t_p\} \text{ and } I_2 = \{t_{p+1}, t_{p+2}, \ldots, t_m\} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (14)$$

Let the first clone of MA be denoted as C_1, and the second clone is called C_2. Hence, C_1 visits sub-itinerary I_1 and C_2 visits I_2. If (m-p) > p, C_2 will be cloned into C_3 and dispatched to node $T_{2p+1,}$ the start point of itinerary I_3.

4. Mobile Agent Architecture

The architecture adopted in this research is shown in Figure 2. The components of the architecture include the following: MA Generator, MA Library, Clone Manager, MA Launcher, MA Disposer, Data Assembler, and Report Library. The description of each module and their specification and design are presented in the following sub-sections.

4.1 Agent Generator

MA Generator is the component that produces the mobile agents for management functions. Any agent generated can be stored in template form in the MA Library for later retrieval. When the network administrator requests the service of an agent, MA Generator checks the MA Library to see if the template for such agent is available and retrieves it if it exists, otherwise it is created.

4.2 MA Library

MA Library serves as the repository for the mobile agents that are designed to perform specific management functions. Any agent generated can be stored in template form in the MA Library for later retrieval, activation, and cloning. When the network administrator requests the service of an agent, the MA Generator checks the MA Library to see if the template for such agent is available and retrieves it if it exists, otherwise it is created.

4.3 Clone Manager

Clone Manager makes copy or copies of the mobile agent and gets it set for activation and onward transmission. To dispatch a single agent only one clone is made while for multiple agents, several clones are made. Even when several clones are made, the administrator decides how many targets each cloned agent should visit but it is at the discretion of each agent to determine which node to visit first and which to visit next. For instance, if the first node is down, the agent can visit the next node first and later return to visit the first node.

When an agent is cloned, a replica of the agent is created and given a unique name. In this work, we make clones for two principal reasons. One is to ensure that original agent is protected and secured. Therefore, instead of sending out the original agent, we send a clone. In the event that the cloned copy is compromised through any security threats, the master agent can be cloned again and dispatched to the target nodes. Even when a single agent is being sent out to visit all managed nodes, a clone may be sent out instead of the master. The second reason is to take advantage of parallelism. When a large number of hosts are to be visited, we can save time by using multiple clones to visit a few nodes. This reduces the turnaround time.

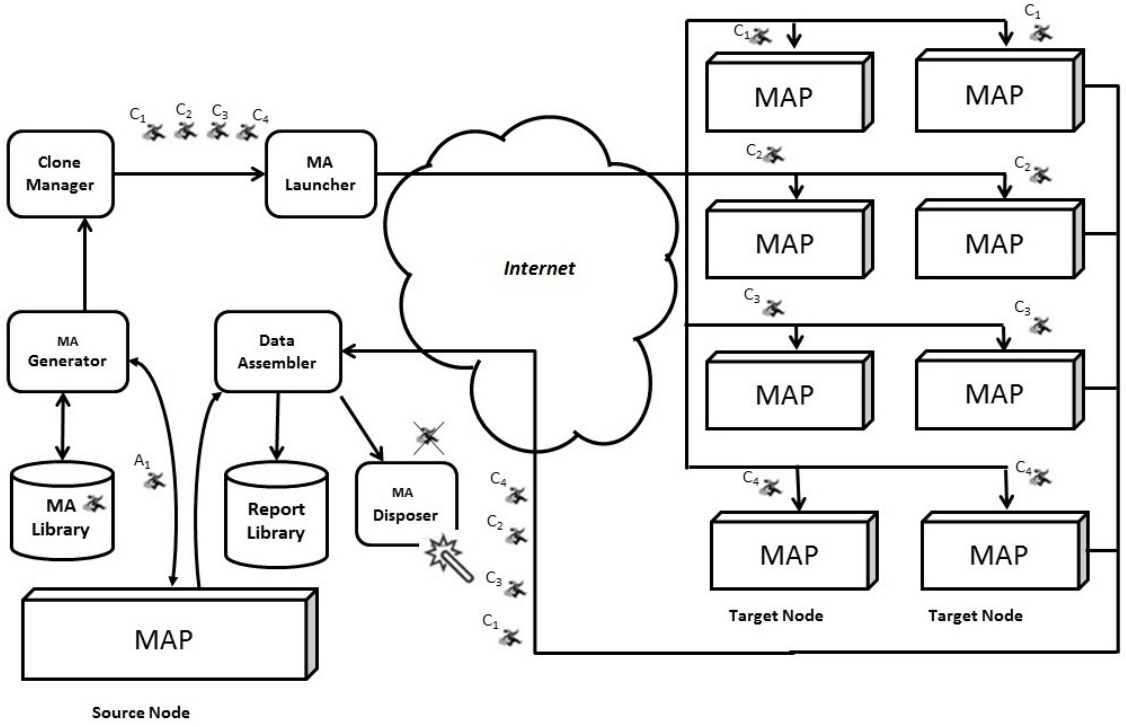

Figure 2 Mobile Agent Architecture

Cloning facilitates fast retrieval of data from managed hosts because the mobile agent can migrate to

multiple hosts simultaneously and take advantage of parallelism. This should be more apparent when dealing with large number of hosts.

4.4 MA Launcher

MA Launcher provides the itinerary or destination address for the mobile agent. MA launcher also serialises the mobile agent and send it to the target node.

In our framework, we create agents to visit nodes and collect management data. Agent must be empowered to visit certain places. They must hence have the address of the places to visit, which we call the itinerary. When a single agent is to visit multiple hosts, the addresses of the hosts are given in the itinerary. When multiple agents are to visit one host each, then they must be dispatched to a particular address. During mobile agent migration from source host to destination host, the design approach adopted is the push migration strategy. In this case, the agent code, its data and state are serialised and migrated at the same time. On getting to the destination host, mobile agent is deserialized and made to resume execution (Figure 3).

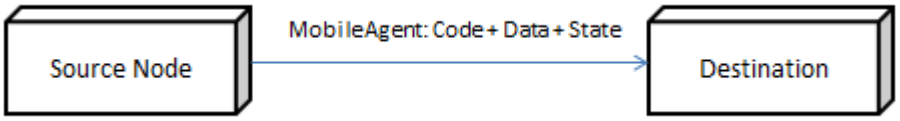

Figure 3 The Migration Strategy

The migration process is depicted in Figure 4. At the source host, the mobile agent is serialised and sent. At the destination, the agent is deserialised and instantiated.

4.5 MA Data Assembler

After interacting with each target host to obtain the necessary network management data, the results obtained by the mobile agent should be assembled

4.6 Report Library

Report Library is a repository where data collected are stored. From here the network administrator can aggregate his report, make analysis and take informed decisions.

4.7 MA Disposer

This component disposes off any agent that is no longer needed in the system. Agent disposal is also called agent dump or agent deletion. As a rule, mobile agent cannot be left to roam aimlessly in the

network as this can be dangerous or at least compete for network resources. Of course, mobile agent templates kept in the MA library do not exist in activated form and pose no dangers for to the network.

5. Conclusion

In this paper, we have described the design of a mobile agent system for monitoring and evaluation of the activities of users in a network environment. A mobile agent can be launched to visit a number of hosts in the network with the view to monitoring their usage of resources within the system. Heavy usage of the Internet and downloads of heavy files and graphics by some users for instance can impact negatively on Internet speed by other members who would love to be on the Internet for routine activities such as checking mails and sending and receiving regular sized files. In these days of multiprogramming operating systems, users may simply open several applications which can consume a lot of the physical memory space, thus reducing the performance of their local system. Downloads from the Internet can also consume a lot of bandwidth and deny other people from smooth assess to bandwidth usage. The proposed system generates statistics on the activities of each user's node on a corporate network which can help the system administrator to evaluate the level of usage of each node on the network to ensure optimal performance. The system may thus serve as a veritable tool to enhance system efficiency and network reliability. In future research, we shall attempt develop a practical realisation of the proposed system and carry out case studies of computer networks.

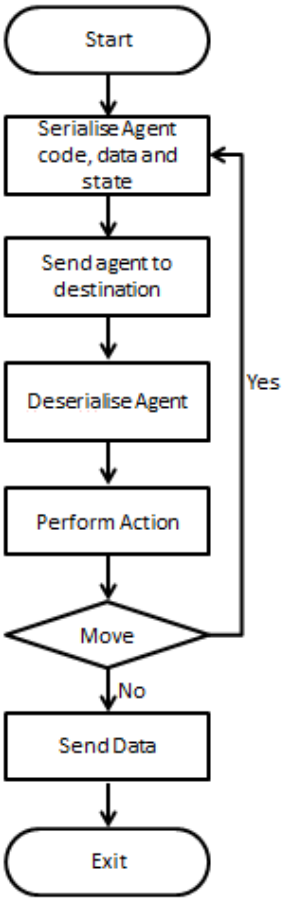

Figure 4 Mobile Agent Migration Process

References

Akinyokun, O.C., and A.A. Imianvan. "Mobile Agent System for Computer Network Management." *International Conference on Advances in Engineering and Technology*. 2009. 796-808.

Al-Kasassbeh, M., and M. Adda. "Analysis of mobile agents in Network fault management." *Journal of Network and Computer Applications* 31 (2007): 699-711.

Boutaba, R., and J. Xiao. "Network Management: State of the Art ." Edited by B.V. Kluwer. *Proceedings of the IFIP 17th World Computer Congress - TC6 Stream on Communication Systems: The State of the Art.* 2002. 127-146.

Braun, P., and W. Rossak. "Mobile Agents: Basic Concepts, Mobility Models, and the Tracy Toolkit." *Morgan Kaufmann,* 2005.

Carzaniga, A., G.P. Picco, and G. Vigna. "Designing Distributed Applications with Mobile Code Paradigms." *In Proc. 19th Int. Conf. on Software Engineering. Boston, Massachusetts*, USA, 1997.

Chen, W.E. and Hu, C. "A mobile agent-based active network architecture for intelligent network control." *Information Science* 14 (2002) 3-35.

El-Gamal, Yousry, Khalid El-Gazzar, and Magdy Saeb. "A comparative performance evaluation model of mobille agent versus remote method invocation for information retrieval." *World Academy of Science, Engineering and Technology* 3 (2007): 286-291.

Gray, R. S., D. Kotz, G. Cybenko, and D. Rus. "Mobile Agents: Motivations and State-of-the-Art Systems."

Technical Report, Dartmouth College, Dartmouth College, 2000.

Imianvan, Anthony Agboizebeta. "Development of a mobile agent for evaluating the use of bandwidth in a computer network." *A PhD Thesis in the Department of Computer Science*, Federal University of Technology, Akure, Nigeria, 2008.

Jain, A. K., M. Aparico, and M. P. Singh. "Agents for process coherence in virtual enterprises." *Communications of the ACM* 42, no. 3 (1999): 62–69.

Jin, G., B. Ahn, and K. D. Lee. "A Fault-Tolerant Protocol for Mobile Agent." *Proceedings of International Conference on Computational Science and Its Applications*. Springer, 2009. 993–1001.

Kotz, D., and R. S. Gray. "Mobile Agents and the Future of the Internet." *ACM SIGOPS Operating Systems Review (ACM)* 33, no. 3 (1999): 7-13.

Lange, D. B., and M. Oshima. "Seven Good Reasons for Mobile Agents." *Communications of the ACM (ACM)* 42, no. 3 (1999): 88-89.

Martin-Flatin, J, and S., Znaty. "Annotated Typology of Distributed Network Management Paradigms." *Technical Report*, Communication Systems Division (SSC), EPFL CH Lausanne, Switzerland, 1997.

Papazoglou, M. P. "Agent-oriented technology in support of e-business." *Communications of the ACM (ACM)* 44, no. 4 (2001): 71–77.

Park, K. "A Fault-Tolerant Mobile Agent Model in Replicated Secure Services." *Proceedings of International Conference Computational Science and Its Applications*. Springer, 2004. 500-509.

Park, T., I. Byun, H Kim, and H. Y. Yeom. "The Performance of Checkpointing and Replication Schemes for Fault Tolerant Mobile Agent Systems." *Proceedings of 21th IEEE Symposium on Reliable Distributed Systems*. IEEE Computer Society, 2002.

Pullum, L. L. "Software Fault Tolerance Techniques and Implementation." *Artech House*, 2001.

Qu, W., H. Shen, and X. Defago. "A Survey of Mobile Agent-Based Fault-Tolerant Technology." *Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies. IEEE*, 2005. 446-450.

Rao, Y and Wang, "Agent-based load balancing routing for LEO satellite networks." *Computer Networks* 54 (2010) 3187–3195.

Shih, T.K. "Mobile agent evolution computing." *Information Sciences* 137, no. 1 (2001): 53–73.

Verma, Vijay K., Ramesh C. Joshi, Bin Xie, and Dharma P. Agrawal. "Combating the bloated state problem in mobile agents based network monitoring applications." *Computer Networks* (Elsevier B.V.) 52 (2008): 3218-3228.