Mobile Agent Framework for Distributed Network Performance Management

Mohamed A. Madkour, Kawther Moria, Fathy E. Eassa, & Kamal M. Jambi
King AbdulAziz University, Faculty of Computing and Information Technology

ABSTRACT
The traditional centralized network management approach presents severe efficiency and scalability limitations in large scale networks. The process of data collection and analysis typically involves huge transfers of management data to the manager which consumes considerable network bandwidth and causes bottlenecks at the manager side. Mobile agent technology provides an effective solution to alleviate this burden by distributing the management functionality over the network elements. A Mobile Agent has the ability to autonomously move among network elements to perform the required tasks locally. Thus, the code is transferred to the data location instead of moving the entire data to the manager's site.

The present study aims to investigate the effectiveness of using mobile agents to overcome the limitations of the centralized structure. Focusing on the network performance management functional area, a prototype is developed to assess the effectiveness of a distributed mobile-agent-based network management system. The developed prototype installs itself automatically on remote machines and periodically checks their software and hardware status. Experiments are done to measure the network traffic volume when managing a typical network. Practical measurements are compared for the traffic generated by both the developed prototype and the current centralized network management standard (SNMP). This comparison confirms that mobile-agent-based management employs much less traffic than the centralized system. An estimation of the required management delays is provided for both sequential- and parallel-dispatching of the mobile agents.

Keywords
Network Management, Network Management Applications, Network Performance Management, Mobile Agents and Agent-based Management

1. INTRODUCTION
Computer networks are indispensible in all aspects of modern civilized life. It is extremely important to have uninterrupted and efficient network operation in order to avoid considerable business losses [1]. Network management plays a central role to achieve this goal. The term Network Management System (NMS) refers to the "sum of all procedures and products for planning, configuring, controlling, monitoring, managing computer networks as well as distributed systems and removing errors in all of these" [2]. NMS's goal is to ensure that network users receive the information system technology services with the quality of services that they expect. To achieve this goal, the International Standard Organization (ISO) defined the OSI network management model that categorizes five areas of function, sometimes referred to as the FCAPS model. These are: Fault Management, Configuration Management, Accounting Management, Performance Management, and Security Management. In the 1990s the International Telecommunication Union (ITU) further refined the FCAPS as part of the Telecommunications Management Network (TMN) recommendation M.3400 on Management Functions [3]. All of these management areas need to collect different management data from remote network elements, using a proper information retrieval process. The present study would focus on the performance management area.

The performance of a given network has a direct effect on the applications that run on that network. Poor network performance badly affects these applications. In some cases, the network may not work at all; in others, it may be slow or unusable. Performance management enables the manager to determine the efficiency of the current network and to develop informed plans for future expansions. In general the performance manager should be able to collect data from key network components about critical performance measures such as the throughput, percentage utilization, packet loss rates and response times. By collecting and analyzing performance data, the network health can be monitored. Trends can indicate capacity or reliability issues before they become service affecting. Performance thresholds can be set in order to trigger an alarm that varies depending upon the severity of the monitored event.

Different management technologies have been proposed to perform the information retrieval process that collects the management data. In the early 1990's two important centralized network management technologies emerged. The Internet Engineering Task Force (IETF) developed the Simple Network Management Protocol (SNMP) [4] which is widely deployed nowadays [5]. Another known centralized NMS is the Common Management Information Protocol (CMIP) developed by the Open Systems Interconnection (OSI) Management System [6]. By the end of the 1990's SNMP is mostly employed for network management because of its simplicity in implementation and use [7].

Both of the systems SNMP and CMIP follow the Manager/Agent structure that uses polling or traps techniques. The management service is done by polling agents to collect required management data. Typically an agent resides at a Network Element (NE). After collecting the required information from all NEs in the managed domain, the manager then aggregates and processes the retrieved data to take whatever necessary actions. The second centralized approach technique is the trap in which an agent that resides at some NE sends a trap message to inform the manager about any critical situation [7].

The rapid growth of networks, the variety of equipments, and the demand for heterogeneity results in a high level of complexity when using the centralized approach to manage very large networks comprising large numbers of NEs. Thus, centralized management approach suffers from problems such as insufficient scalability where the process of data collection and analysis typically involves massive transfers of management data. This scenario then causes extensive strain

on network throughput and processing bottlenecks at the manager machine [8]. Moreover, the centralized approach may suffer from unsatisfactory reliability and flexibility if the management server fails. In such case the entire network would halt until the connections with the server are restored again.

The mobile agent (MA) technology, emerged in the mid 1990's, provides plausible solutions to overcome the limitations of the centralized network management paradigm. Such solutions are based on exploiting the MA's ability to autonomously migrate to different NEs and independently perform complex management tasks. A mobile agent can suspend its execution according to some factors and resume it in another place. Additionally, it has the ability to get the best choice of when, where to go, and what to do according to changes of the network environment [9]. Therefore, MAs offer a powerful management technique represented by the efficient use of the network's bandwidth, and the high degree of flexibility and NE re-configurability. It is believed that MAs can provide better solutions to network performance by saving the need to move large amounts of management data [10, 11, 12, and 13]. In this research, we investigate the possibility of building a mobile-agent-based system to provide performance management functionality in computer networks.

The present work aims to investigate the possibility of using the MA technology to provide network performance management functionality in a real distributed network. A prototype is developed to conduct practical experiments in real network environments to measure and assess the performance of a typical real network. It provides a basis for comparing the suggested approach with the current centralized SNMP approach. The rest of this paper is organized as follows. Section 2 discusses the related work, and Section 3 provides a detailed design of the proposed MA framework. Section 4 provides an overview of the prototype implementation based on the proposed MA framework, presents the experimental work done, and discusses the obtained results. Finally, Section 5 presents the conclusions and gives directions for future work.

2. RELATED WORK

Network performance management should provide solutions to ensure that traffic over the network is effectively managed to optimize performance for all users. It consists of measuring, modeling, planning, and optimizing networks to ensure that they carry traffic with the speed, reliability, and capacity that is appropriate for the nature of the running applications and the cost constraints of the related organizations.

Many researchers conducted their researches to resolve the problem of network management. Hyojoon Kim et al [14] used software defined networking (SDN) paradigm in improving network management. They provided better control over tasks for performing network diagnosis and troubleshooting. The sources performance problems can be determined by network operator.

In [15] Saber Zrelli et al ouline the main challenges that face organizations in managing large scale heterogeneous network infrastructures. They also proposed extended network management framework that is based on service oriented architecture.

In [16], authors proposed a design and implementation of Web-Based Management (WBM) network system. The system include seven modules, one of them is equipment performance management.

Migration is not a new trend; it has surfaced several times in the last three decades. The basic idea is to move some computational structure over the network and execute it remotely. Internet worms, SQL, viruses, mobile objects, postscript, and applets are broadly speaking examples of code that can be moved over the network for remote execution. But the autonomy of MA migration and its reactivity features is really what attracts the attention of many researchers [11].

To overcome the network management complexity resulting from the ever growing networks, the MA technology provided a new management scheme. The motive behind the agent mobility is that, it addresses many limitations faced by the traditional centralized Manager/Agent architecture regarding the information retrieval. In particular it allows minimizing the bandwidth consumption, supporting network load balancing, enhancing scalability and flexibility, increasing fault tolerance and solving problems caused by unreliable network connections.

Several researches incline to integrate the MA technology with the existing network management approaches to adapt with rapidly network changes. Guo et al [17] present a distributed plug-and-play network management model using MAs to implement an automatic and self-adapting NMS. In this model, network components can be automatically located and get initial management tasks without any human intervention. Chou et al [18] designed a Web-based NMS that uses MAs to perform the management tasks. They integrate both the web technology, which offers all the NMS services from the web browser, with the MAs technology.

On the other hand, many researches tend to integrate the MA technology with the SNMP approach being the most NMS used at present. For example, Reuter and Baude [12] developed an integrated architecture using the MA technology together with the SNMP approach to generate an up-to-date network topology. Their developed MA architecture implements the discovery process to record the network topology and refreshed it when needed. This result would be of extreme importance to the network management applications.

Currently, there are intense researches in using the MA technology in the five network management functional areas [3] mentioned in Section 1. In performance management, Guanyu et al [11] propose a MA framework which solves the problem of the information retrieval process that is expressed in the centralized approach. They gave evidence that the network performance can be improved by using MA over a large network size. Gavalas [1] identifies ways for the effective use of the MA technology in distributed management applications, considering large size and dynamic networks. His study addresses a set of requirements for tailoring the MA platforms to meet the management applications, stressing that the MA framework should be implemented to satisfy these requirements. He provided a comparison, regarding network performance, between the centralized management approach and the distributed objects-based approaches. Foster et al [19] describe a powerful decentralized agent control and management strategy in performance tuning that prevents system overflow and maintains good overall system throughput.

Several researches tend to evaluate the mobile agent system performance itself. Xuhui et al [20] built a simulation

environment that evaluates the performance of the MA systems which run on a LAN or in a single host machine. The simulator prototype is developed using the Aglets platform tool [21], and they tested it with certain MA-based algorithms to ensure its functionality.

On the theoretical side, a formal model for the MA systems is provided by Zhen et al [22]. They describe a new formal language for distributed network management based on mobile agents. Moreover, a New MA communication protocol is offered by Bernich and Mourlin [23] with a flexible scheme to exchange data between MAs.

3. THE PROPOSED MA FRAMEWORK

This section presents detailed design of a proposed Mobile Agent System (MAS) for enhancing the performance of retrieving management information and managing the performance of computer networks. This approach alleviates network performance problems faced by almost all network administrators when using the current centralized network management standard, namely SNMP. In SNMP the process of retrieving management information suffers from latency and network bottleneck on the manager's side. This is because the manager is responsible for collecting huge dynamic and duplicated information, analyzing it locally, and using it in the management applications.

A prototype is developed according to the proposed MAS framework, and practical experiments are conducted to assess its performance and compare it with that of the SNMP standard.

3.1 The MAS Architecture

Figure 1 shows the main components in the developed MAS. This system employs several interacting static and mobile agents, a global database, and one local database in each host. The agents cooperate by exchanging messages to collect data and compute several performance measurement variables such as the number of dropped packets, the number of errors in the packets, the use of the ICMP messages, link and device utilization, and etc.

The following is a brief description of the MAS components:

1. The ***MainAgent*** is a stationary agent that controls the entire MAS system. Once it is installed, it creates all needed classes at the local manager and dispatches a set of mobile ***InstallationAgents*** to a list of addresses, belonging to the managed hosts and devices, to start the installation process.

2. The ***InterfaceAgent*** is a stationary agent that resides at the manager machine and works as a system interface to the administrator. It accepts input values that the system needs, such as addresses of all the connected hosts, and passes them to the ***MainAgent***.

3. The ***InstallationAgent*** is a mobile agent that is dispatched to all hosts in the managed network to remotely install the MAS system at every host.

4. The ***CheckAgent*** is a stationary agent that resides at the manager machine and checks the history database for hardware and software status. This agent produces periodic status reports, for example monthly or weekly. The network's administrator examines these reports to identify any improper states in the managed hardware and/or software components.

5. The ***GeneratorAgent*** is a stationary agent that generates the MAs and dispatches them to their destinations. It gets the required addresses from the ***MainAgent***.

6. The ***IntegrityMobileAgent*** is a mobile agent that regularly visits all managed elements in the network to check the integrity of the MAS.

7. The ***CheckUpAgent*** is a mobile agent that collects updated information from the hosts on the network then returns back to the manager to update the database accordingly.

8. The ***SNMP-Agent***: In SNMP the "Management Information Bases" (MIBs) describe the structure of the management data of a device subsystem using a hierarchical namespace containing object identifiers (OID). Each OID identifies a variable that can be read or set via SNMP. The ***SNMP-Agent*** is a stationary agent that reads the required OID variables. It then sends the results locally to the ***ObserverAgent***.

9. The ***ObserverAgent*** is a stationary agent responsible of observing the hardware and software status of all managed network elements. Once the OID values are sent from the ***SNMP-Agent***, it starts reading and analyzing them. Upon detecting any error, the ***ObserverAgent*** sends a message to the ***MainAgent*** containing the error details.

10. The ***PerfAgent*** is a mobile agent dispatched to every host to perform the performance measurement calculations.

11. The ***History Database*** is a repository of history information about each and every piece of software and hardware installed in the managed network. The collection of history information for each network element starts from the time that it's installation.

12. The ***Host Database***: Every host has its own ***Host Database*** that stores the latest updated MIB values as retrieved by the ***ObserverAgent***.

3.2 System Design

MAS has two main tasks, namely to install itself automatically on a host machine and to collect network management data from remote network elements (hosts and etc.). This data is eventually analyzed at the system manager to evaluate the network's performance.

3.2.1 Automated System Installation

MAS system installs its components at all hosts connected to the network. Once the system automatically starts up, this process is done independently without any human intervention. Figure 1 shows the process of creating the MAS agents in two hosts. The process starts with loading the ***InterfaceAgent***. It creates the ***MainAgent*** which creates the ***GeneratorAgent*** and *the History database*. All these agents are stationary agents that reside at the manager machine.

The automatic installation process proceeds as follows at every host:

1. The user passes the list of host addresses, as Aglet addresses, to the ***InterfaceAgent***. These addresses are then compiled as an array of strings and sent to the ***MainAgent***. The ***MainAgent*** sends the message "Install" to the ***GeneratorAgent***. It contains the route scheme (parallel, sequential or SiG) and the address list.

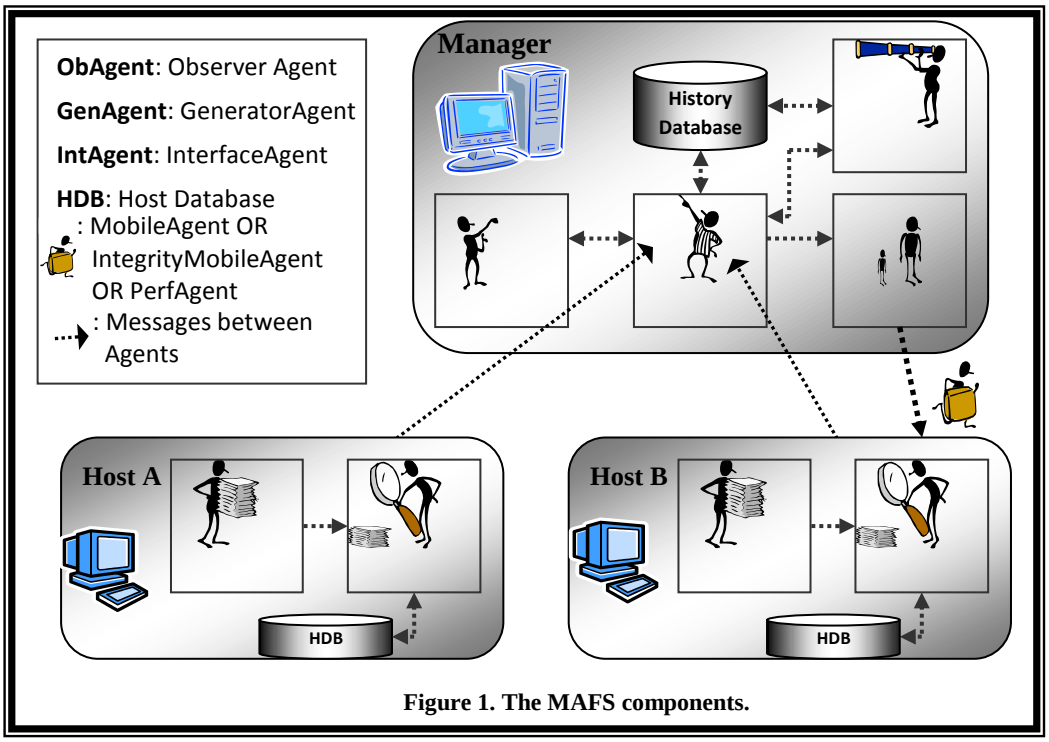

Figure 1. The MAFS components.

2. The ***GeneratorAgent*** creates instances of the ***InstallationAgent*** and dispatches them in parallel one instance to each host.

3. Upon arriving at its target host, each instance of the ***InstallationAgent*** starts the initialization process. This results in creating the host database file and extracting the MAS agents and their objects.

4. Next, it will run the ***ObserverAgent*** and the ***SNMP-Agent*** to start the MAS system. Both agents will collaborate to collect the entire host information and then pass the collected data to the ***InstallationAgent***.

5. Next the ***InstallationAgent*** would return back to the manager with the results and then dispose itself.

A confirmation is needed to check if the installation is done correctly. Within a specified timeout period, the manager must receive an acknowledgement from every host on the list confirming proper installation. If any host fails to respond within the timeout, this means that either the ***InstallationAgent*** was totally lost or the installation process failed. Consequently the ***MainAgent*** must resend another instance of the ***InstallationAgent***. Should that failure persists then the administrator should be informed to start human intervention.

3.2.2 History and Host Databases

The "History" is a global database used in the MAS system to store the collected management data from all hosts in the system. Typically the manager records the entire OID variables that represent host specifications for every host. These data are collected periodically and stored in the "History" database indexed by the date and time of collection. Figure 2 shows the structure of this database. The second database type used in the MAS system is the HostDatabase.

Every host in the system has its unique local HostDatabase that stores the most up-to-date information about the host hardware and software specifications.

It also stores the OID variables used in the status detection process. Figure 3 shows the structure of this database.

3.2.3 Performance Measurements

The MainAgent instructs the GeneratorAgent to generate the PerfAgent and let it visit every host to perform host utilization performance measurement calculations. At every visited host, the PerfAgent will compute the percentage utilization of the network interface over two consecutive time instances, "X" and "Y". The following outlines the PerfAgent actions done at every host.

1. Send "HR-OID" message to the SNMP-Agent to get the values of the required OID variables, as shown in Table-1, for the instance "X".
2. After some delay, say 50 seconds, repeat step 1 above to get the values of the same OID variables for the instance "Y".
3. After performing the required calculations, as shown in Table-2 [24], the PerfAgent would dispatch itself to the manager as shown in Figure 4.
4. Once returned back to the manager, it will store the obtained results in the "History" database and then passes them to the InterfaceAgent who displays them to the administrator.

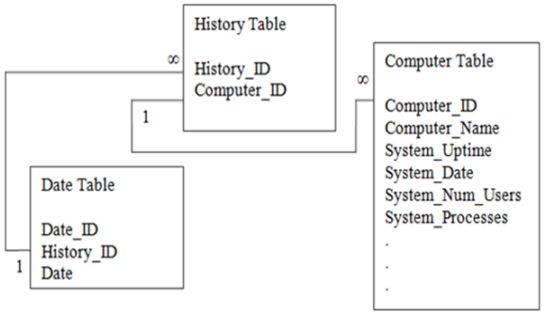

Figure 2. Structure of the "History" database

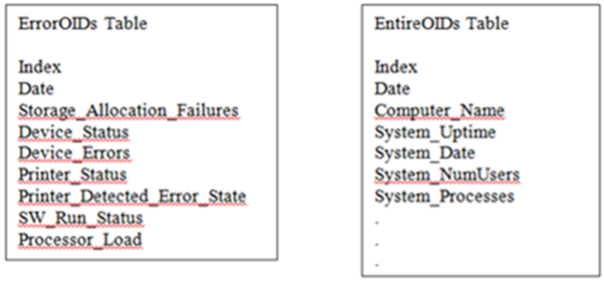

Figure 3. Structure of the "Host" database

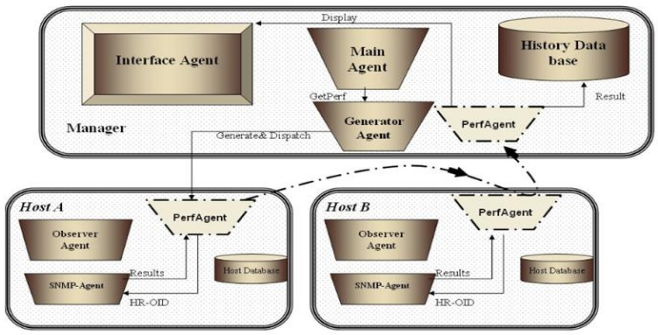

Figure 4. The Performance Management Process

Table 1. OID variables defined in RFC 1213 MIB

OID variables	Meaning
ifInOctets	The total number of bytes received on the interface
ifOutOctets	The total number of bytes transmitted out of the interface
ifSpeed	The estimation of the interface's current bandwidth in bits per second

Table 2. Variables for computing the utilization

Variables	Meaning
ifInOctetsx	The value of ifInOctets that is measured at instance "X"
ifInOctetsy	The value of ifInOctets that is measured at instance "Y"
ifOutOctetsx	The value of ifOutOctets that is measured at instance "X"
ifOutOctetsy	The value of ifOutOctets that is measured at instance "Y"

The utilization of the network interface is computed as follows in equation 1.

Length of the polling period= Y - X

Total bytes = (ifInOctetsy– ifInOctetsx) + (ifOutOctetsy – ifOutOctetsx)

Total bytes per sec = Total bytes / (Y-X)

Utilization = (Total bytes per sec * 8) / ifSpeed (1)

4. TESTING AND EVALUATION

This section explains the structure of the MAS system. A prototype is implemented using JAVA under management of the IBM Aglets framework. Several experiments are conducted to evaluate the MAS performance in retrieving the MIB tables. The MAS system uses its "CheckUpAgent" to collect the updated management information from a group of hosts and then store it in the "History" database.

The experiments compare the standard SNMP approach with the developed MAS approach. Two performance measures are considered in this regard, (a) measuring the network traffic to determine the scalability, and (b) estimating the management time.

4.1 Comparison of the Network Traffic

Two experiments measure the network traffic generated to retrieve required data from the MIB database in both cases. Two hosts are used; "Host A" runs as a manager and "Host B" runs as the managed node. A Graphical User Interface tool, Wireshark Network Protocol Analyzer, is used to capture the flow of packets and monitor the traffic. This tool allows to interactively browsing packet data from a live network.

Experiment 1: The AdventNet SNMP Agent is used as a centralized manager. It runs on "Host A" and polls the entire MIB table (Host Resources MIB) from "Host B". The entire MIB tables that consist of around 64 OID are fully transferred, however only few rows are needed.

During the operation of transferring the required information, the Wireshark tool has captured the network traffic and determined how many packets were exchanged between both hosts. Figure 5 shows the obtained results. A total of 519 packets were captured for transferring the entire MIB tables, which is about 750 kilo bytes.

Experiment 2: In the MAS approach, the CheckUpAgent moves from "Host A" to "Host B" to get the required data, which is around 10 OID values, and returns back. There is no need to get the entire MIB database, only the updated information. During this operation the traffic generated between both hosts is captured by the Wireshark tool. Figure 6 shows that the data retrieval is completed using 11 packets, which is about 740 bytes.

As shown in Table-3, the obtained results are clearly in favor of the MAS system with a total traffic volume being three orders of magnitude less than the traffic for the centralized SNMP management. It should be noted however that the results of Experiment 2 may differ depending on the amount of changes occurred in the required management information. In fact, more changes in the software and hardware specifications of the managed hosts would slightly increase the transferred data.

Figure 5- Results of Experiment 1

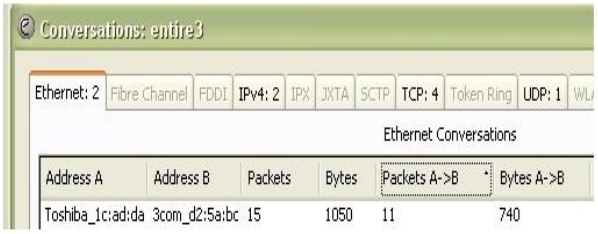

Figure 6- Results of Experiment 2

Table 3. Network traffic comparison

Network Manager	Traffic Type	No. of Packets Exchanged	Total Bytes Exchanged
Standard SNMP	Retrieving the entire MIB table; 64 OID's	519	750 kByte
Developed MAS	Retrieving the required information; 10 OID's	11	740 Bytes

4.2 Management Time Estimation

Table-4 shows the variables used to estimate the management time for the developed mobile-agent-based MAS system. The total delay to retrieve the required MIB information is considered here as a measure of performance. This is computed as the transmission delays of the mobile agent that collects the management data plus the time required to collect that data from the visited hosts

In computing the total time delay for the MAS system two dispatching schemes are considered, namely sequential dispatching and parallel dispatching as shown in Figure 7 and Figure 8, respectively.

Table 4. Variables for the Time Response Computations

N	The number of hosts that the mobile agent visits.
Th_i	The time that the CheckUpAgent consumes to collect management data at the ith host.
Th_s	The total time that the CheckUpAgent consumes to collect management data from all N hosts using sequential dispatching.
Tx_i	The transmission time to dispatch the CheckUpAgent to the ith host from the previous host in its itinerary.
Tx_s	The total transmission time consumed to dispatch the CheckUpAgent from the manager host to all N hosts using sequential dispatching and returning back to the manager.
Ts	The total round trip delays of the CheckUpAgent to visit N hosts using sequential dispatching.
Tp	The total round trip delays of the CheckUpAgent to visit N hosts using parallel dispatching.

4.2.1 Delay Time in the Sequential Dispatching Scheme

The total time that the CheckUpAgent consumes to perform its management tasks at all N hosts using sequential dispatching is the sum of computation delay at the managed hosts plus the transmission delays for dispatching the agent, as given by equation 2.

$$Ts = Th\_s + Tx\_s \qquad (2)$$

The management information collected by the CheckUpAgent depends on the hardware and software specifications of the visited hosts. An experiment has been done using two machines Host A and Host B. Host A contains few (17) installed software applications, where the host B contains a large number (118) of installed applications. A timer is used to measure the time that the CheckUpAgent spends in each host to collect the management information. Actually, 10 experiments have been done at each tested machine and the average over these results is taken to set an upper and lower bounds on the time that the CheckUpAgent spends in an arbitrary host to collect the management information. The computed upper and lower bounds are 1250.2 ms and 3054.5 ms, respectively. The delay time at the ith host will be estimated as a random value Th_i between the two bounds.

Consequently, the total time spent at N hosts is estimated as follows:

$$Th\_i = Random(1250.2, 3054.5) \qquad (3)$$

$$Th\_s = \sum_{i=1}^{i=N} Th\_i \qquad (4)$$

To estimate the network transmission delay, a simple experiment is conducted to ping multiple random sites in the Internet to get upper and lower bounds of the transmission delays. Ten experiments are carried out at different times of the day to ping a large set of about 30 sites selected at random in each experiment. The average values of the transmission delay bounds are found to be 1.1 ms and 61.2 ms, respectively.

In sequential dispatching, the CheckUpAgent visits N hosts and returns back to the manager. The delay for each of these N+1 transmissions is considered to be a random value between the obtained upper and lower bounds. The total transmission delay is estimated as follows:

$$Tx\_i = Random(1.1, 61.2) \qquad (5)$$

$$Tx\_s = \sum_{i=1}^{i=N+1} Tx\_i \qquad (6)$$

4.2.2 Delay Time in the Parallel Dispatching Scheme

In this scheme each instance of the CheckUpAgent would be dispatched to one host machine and then returns back with the result. Following the same argument in the sequential dispatching, the total delay time is given by equation (7).

$$Tp = Random(1.1, 61.2) + Random(1250.2, 3054.5) + Random(1.1, 61.2) \qquad (7)$$

Figure 9 shows the sequential and parallel management delays for up to 12 managed hosts. The results are obvious but it is presented here to give numerical estimation of the total network management delay in both sequential- and parallel-dispatching schemes.

As shown in Figure 9, the management delay in case of sequential dispatching is almost linearly proportional to the number of managed hosts. A hybrid dispatching scheme would be advantageously employed to limit the management delay should the number of hosts be extremely large. In such

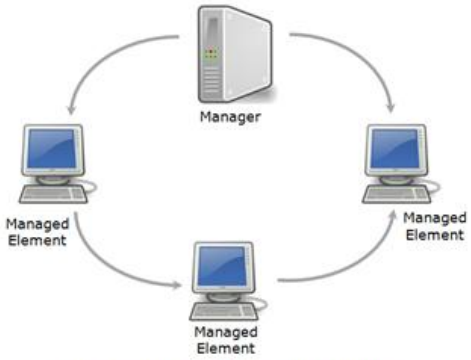

Figure 7- Sequencial Dispatching Scheme

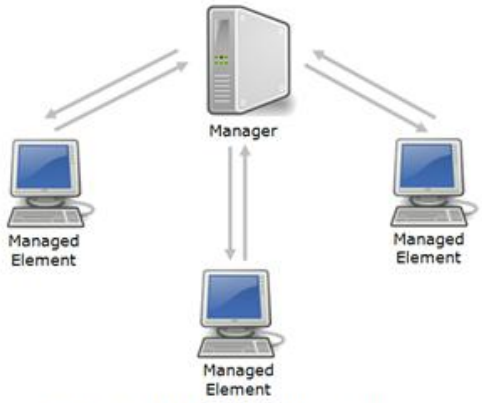

Figure 8- Parallel Dispatching Scheme

scheme the managed hosts will be divided into several groups of about 10 hosts each. Each group will be managed sequentially by a different instance of the CheckUpAgent. The exact number of hosts in a group depends on a trade-off. On one hand, the management delays will increase for a large group size. On the other hand, in case of small groups the total number of trips done by the CheckUpAgent instances will increase. This will directly increase the required network bandwidth. Moreover the manager overhead to handle many instances of the CheckUpAgent will also increase.

5. CONCLUSIONS

The present work investigates the effectiveness of using mobile agent technology for the management of computer networks compared to the centralized technique used in the SNMP standard. In particular, the mobile agent technology introduces significant advantages that enhance the network management applications performance. It automates the management functions, reduces network traffic, distributes the management workload, supports for heterogeneous environments, and supports on-line extensibility of services. Moreover, it has the ability to work off-line autonomously, it is easy to upgrade, and provide many off-line fault tolerance services.

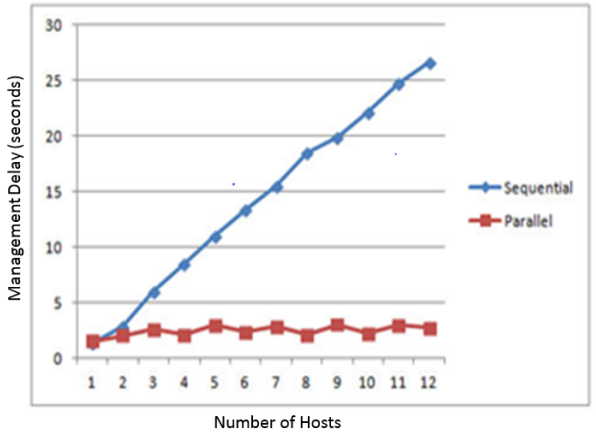

Figure 9. Estimated Management Delays for sequential and Parallel Dispatching

Both sequential and parallel dispatching of the mobile agent to the managed network element (hosts and etc.) is considered here. The main issue that affects mobile agent systems is the mobile agent's size, the smaller the better in order to keep its transmission time low. Towards this goal the initial mobile agent code should be minimized by storing part of the management functionality at the managed network element. Secondly, to limit the size of its memory buffers the parallel dispatching scheme seems appropriate. However, this would lead to an increase in the required network bandwidth should the number of managed elements becomes excessively large. A hybrid dispatching scheme would be more appropriate in this case.

The present work provides a framework for using a mobile agent system (MAS) in network management, and gives an outline for the system's design. A prototype is developed and used to conduct experiments in real network environment and compare its performance, regarding the required traffic volume, to the Internet's standard centralized SNMP management protocol. The results indicated a great reduction in the traffic volume in favor of the developed system. An estimation of the management delay is also investigated and a numerical comparison between parallel and sequential dispatching schemes is provided.

A final word must be stated here regarding the implementation of mobile agents in real life networks and in the Internet. Such systems still require greater effort to implement it practically because the existing platforms provide only basic mobility functions. More importantly they suffer from security threats, and need to be tailored for the management functions. These problems need further attention to allow for safe and effective implementation of the mobile agent technology in managing computer networks and internets.

6. Acknowledgement

The members of the research team are thankful to King Abdulaziz City for Science and Technology (KACST) for their grant (AT-30-114) for this project.

7. REFERENCES

[1] D. Gavalas, Mobile Software Agents for Network Monitoring and Performance Management, Ph.D. Thesis, University of Essex, UK, July 2001.

[2] G.Hegering, S. Abeck, Integrated Network and System Management, Addison-Wesley, 1994.

[3] ITU-T recommendation M.3400 (02/2000).

[4] J. Case, M. Fedor, M. Schoffstall, J. Davin, A Simple Network Management Protocol (SNMP), RFC 1157, May 1990.

[5] Z. Hong, G. Feng, Y. qiang, W. Xing, Research on Mobile Agent-based Hierarchical Network Management Model, Proceedings of IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications (MAPE), 2005.

[6] Information Technology, Open Systems Interconnection, Common Management Information Protocol (CMIP) – Part 1: Specification, 1998.

[7] M. Subramanian, Network Management Principals and Practice, Addison Wesley, 2000.

[8] S. Tokdemir, Survey paper: mobile agents – architecture and applications, spring 2005.

[9] V. Pham, A. Karmouch, Mobile Software Agents: An Overview, IEEE Communications Magazine, pp. 26-37, 1998.

[10] A. Michalas, Enhancing The Performance Of Mobile Agent Based Network Management Applications, Proceedings of Sixth IEEE Symposium on Computers and Communications (ISCC'01) pp. 0432, 2001.

[11] L. Guanyu ; W. Baofeng; Y. Yang; A. Lihua , Researches on Performance Optimization of Distributed Integrated System Based on Mobile Agent, The Sixth World Congress on Intelligent Control and Automation, pp. 4038- 4041, June 2006.

[12] E. Reuter, F. Baude, A mobile-agent and SNMP based management platform built with the Java ProActive library, IEEE Workshop on IP Operations and Management, pp 140- 145, 2002.

[13] L. Jing-hua, X. Guang-hui, A New Network Management Framework Design and Application Realization, In Proceedings of the Sixth international Conference on Parallel and Distributed Computing Applications and Technologies, IEEE Computer Society, 2005.

[14] Hyojoon Kim and Nick Feamster "Improving Network Management with Software Defined Networking " IEEE Communications Magazine, February 2013.

[15] Saber Zrelli, Atsushi Ishida, Nobuo Okabe and Fumio Teraoka "ENM: A Service Oriented Architecture for Ontology-Driven Network Management in Heterogeneous Network Infrastructures" IEEE/IFIP 4th Workshop on Management of the future Internet, 2012.

[16] Liu ZhiLong, Chun Hu, and Wei Ge. "WBM based network management system design and Implementation" 9th International conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012).

[17] Nan Guo, Tianhan Gao, and Hong Zhao, Distributed Plug-and-Play Network Management Model Based on Mobile Agents, Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE '04), 28-31 March 2004 , pp. 487 – 491.

[18] L. Chou, K. Tang, C. Kao, Multiple/mobile-agent-based network management systems for Taiwan's National Broadband Experimental Networks, Global Telecommunications Conference, IEEE Computer Society pp. 1975- 1979 , November 2002.

[19] S. Foster, A. Nebesh, D. Moore, J. Flester, Performance Tuning Mobile Agent Workflow Applications, In Proceedings of the Technology of Object-Oriented Languages and Systems, IEEE Computer Society, 1999.

[20] L. Xuhui, C. Jiannong, H. Yanxiang, C. Yifeng, MADESE: A Simulation Environment for Mobile Agent, Proceedings of CIT 06, IEEE Computer Society, 2006.

[21] Bill Venners, "The architecture of aglets", JavaWorld.com, 04/01/97.

[22] H. Zhen, W. Zhen, L. Xiao, Formal Language Description of Mobile Agent's Theory Model, Proceedings of Machine Learning and Cybernetics International Conference ,pp. 185-187, 2006.

[23] M. Bernich, F. Mourlin, "Mobile agent communication scheme", International Conference on Systems and Networks Communication (ICSNC '06), pp. 6-6, October 2006.

[24] I. Adhicandra , C. Pattinson, "Performance Evaluation of Network Management Operations", Proceeding of 3rd Annual Symposium of Postgraduate Networking Conference(PGNET), pp. 210-214, 2002.