Scalability of a Mobile Agents Based Network Management Application

Marcelo G. Rubinstein, Otto Carlos M. B. Duarte, and Guy Pujolle

Abstract: **This paper analyzes mobile agent performance in network management compared to the client-server model used in the Simple Network Management Protocol (SNMP). Prototypes of an application that gathers MIB-II (Management Information Base-II) variables have been created and tested on a LAN. After acquiring implementation parameters related to network management and to the mobile agent infrastructure, simulation results have been obtained on large topologies similar in shape to the Internet. Response time results show that mobile agents perform better than SNMP when the number of managed elements ranges between two specific limits, an inferior bound and a superior one, determined by the number of messages that pass through a backbone and by the mobile agent size which grows along with MIB-II variables collected on network elements. The results also show that a significant improvement is achieved when the mobile agent returns or sends data to the management station after visiting a fixed number of nodes.**

Index Terms: **Mobile agents, network management, performance evaluation.**

I. INTRODUCTION

Most network management systems are based on the client-server centralized paradigm. Management implies access to a large quantity of dynamic network information, which is collected through periodic polling. This fine grained client-server interaction generates an intense traffic that overloads the management station [1], resulting in serious scalability problems.

Some decentralization steps have already been taken. In event notification, SNMP (Simple Network Management Protocol) agents notify the management station upon the occurrence of a few significant events. These agents use traps, i.e., messages sent without an explicit request from the management station, to decrease the intensive use of polling.

A more decentralized approach is adopted in SNMPv2 (SNMP version 2), in which there are multiple top-level management stations known as management servers. Each such server is responsible for managing a set of agents. Nevertheless, it can delegate responsibility to some intermediate manager. This manager, called a proxy agent, monitors and controls agents under its responsibility. It also works as an informative agent and is controllable by a higher-level management server.

Manuscript received March 22, 2002; approved for publication by Raouf Boutaba, Division III Editor, June 11, 2003.

M. G. Rubinstein is with the Departamento de Engenharia Eletrônica e Telecomunicações, Universidade do Estado do Rio de Janeiro, Brazil, email: rubi@uerj.br.

O. C. M. B. Duarte is with the Grupo de Teleinformática e Automação, Universidade Federal do Rio de Janeiro, Brazil, email: otto@gta.ufrj.br.

G. Pujolle is with the Laboratoire LIP6-CNRS, Université Pierre et Marie Curie, France, email: Guy.Pujolle@lip6.fr.

RMON (Remote MONitoring) uses network monitoring devices called monitors or probes to perform proactive LAN monitoring on local or remote segments. These probes provide information about links, connections between stations, traffic patterns, and status of network nodes. They also detect failures, misbehaviors, and identify complex events even when not in contact with the management station.

These proposals seem to reduce traffic at the management station. But, as the computational power of network nodes is increasing, it is possible to delegate more complex management functions to these nodes. Moreover, in order to satisfy the multiple requirements of today's networks, novel network management systems which can analyze data, decide, and take proactive measures to maintain the Quality of Service (QoS) of the network must be developed [2]. Mobile agents are a good alternative to satisfy these needs.

Network management could be distributed and made scalable through mobile agent technology. These agents are programs that help users by acting on their behalf in performing a number of tasks in the network. These agents move to the place where data are stored and get information the user needs. They help saving bandwidth, time, and money. Mobile agents decentralize processing and control. As a consequence, they reduce traffic around the management station and makes asynchronous the agent-manager interaction. This is very useful in the presence of unreliable or lossy links. They also distribute processing load, and increase flexibility by allowing the modification of the management agents' behavior. Managers can also delegate authority to mobile agents by decentralizing management functions such as local data processing and filtering. In addition, several agents may be used to manage a complex network. They can negotiate the partitioning of the network to be managed and can avoid redundant searches through exchanging information they already have with other agents.

Research activities related to mobile agents in network management are recent [3], [4]. Several researchers investigate the performance of network management based on mobile agents. Geihs *et al.* [5], El-Darieby and Bieszcad [6], Outtgarts *et al.* [7], and Picco [8] provide simple quantitative evaluation of mobile agent and client-server approaches. Puliafito *et al.* [9], Baldi *et al.* [1], and Liotta *et al.* [10] present a performance comparison based on more sophisticated mathematical models. These models, however, do not consider parameters related to network management and to the agents infrastructure, which are vital for obtaining more reliable results pertaining to a real implementation. Bohoris *et al.* [11], Gavalas *et al.* [12], and Sahai and Morin [13] perform measurements of mobile agent performance but only on LANs of a few nodes. The problem of scalability of network management based on mobile agents using real imple-

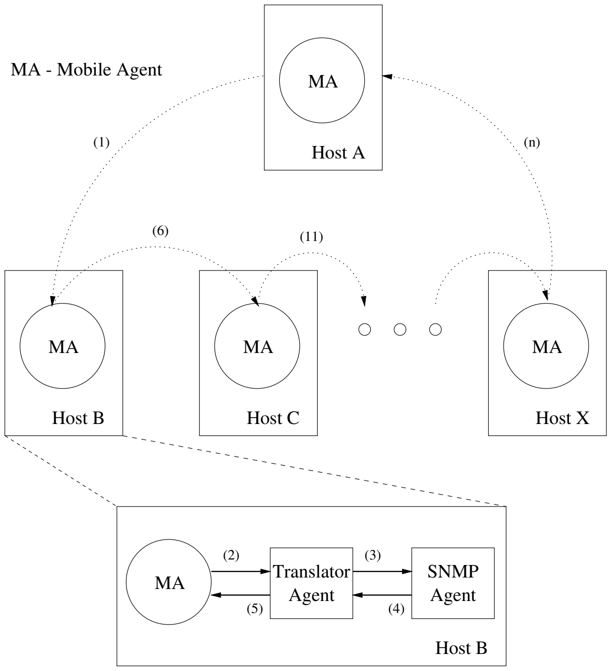


Fig. 1. Network management by using a mobile agent.

mentation parameters on a complex Internet-like network with a high number of nodes has not been exhaustively explored yet.

In this paper, we compare the scalability of network management based on mobile agents against traditional SNMP based management through the analysis of some simulation and implementation results. Two application prototypes that gather MIB-II (Management Information Base-II) variables have been created and tested on a LAN. The first one is based on mobile agents and the latter only uses SNMP. After acquiring parameters related to network management and the mobile agent infrastructure, new results are obtained on large topologies similar in shape to the Internet. Moreover, we also evaluate the effects on network management performance of the MIB-II variable to be gathered, of the transport protocol used to send the mobile agent and SNMP data, and of returning or sending data to the management station.

This paper is organized as follows. Section II presents the implemented prototypes and measurement results. Section III reports simulation results. Finally, concluding remarks and future directions are presented in Section IV.

II. IMPLEMENTATION OF A MANAGEMENT APPLICATION

Since SNMPv2 is not as widespread as SNMPv1, which does not scale to large complex networks, we use two different solutions for gathering MIB-II variables on managed elements: a mobile agent-based solution and an SNMP based one.

The Mole infrastructure version 3.0 [14] is used in the mobile agent implementation. Mole has been the first one to use Java and has been chosen because it is free and open source. Mole provides the functionality for the agents to move, communicate with each other, and interact with the underlying computer system. It uses TCP to transfer them. The agent's data contain global and instantiated variables. A weak migration scheme is provided where only the state related to these data is transferred. As a consequence, the programmer is responsible for encoding program variables in the agent's execution state. An execution state includes local variables, parameters, and execution threads.

Both of the implemented prototypes use SNMP to gather MIB-II variables. The AdventNet SNMP library [15] and the snmpd from package ucd-snmp [16] have been used. The AdventNet SNMP package contains APIs to help the implementation of solutions and products for network management. Version 2.2 of the AdventNet SNMPv1 has been used. The daemon snmpd, which comes with Linux Red Hat, is an SNMP agent that responds to SNMP request packets. The package versions that have been used on this experiment are the 3.5.3, for machines running the Red Hat 5.2, and the 4.0.1 for the Red Hat 6.x.

The mobile agent implementation (Fig. 1) consists of one mobile agent, a set of SNMP agents, and a set of translator agents. The mobile agent migrates to all of the network elements that need to be managed. The SNMP agents access the MIB-II variables. The translator agents convert the mobile agent requests into SNMP requests[1]. There is exactly one SNMP agent and one translator agent per network element. The mobile agent migrates to a network element (arc 1 of Fig. 1) and communicates by Remote Procedure Call (RPC) with the translator agent (arc 2). This translator agent sends a request (GetRequest PDU of SNMP) to the SNMP agent (arc 3) and obtains the response (arc 4), which is sent to the mobile agent (arc 5). Then, the mobile agent goes to the next element (arc 6) and restarts its execution. After finishing the task of visiting all network elements that need to be managed, the mobile agent returns to the management station (arc n).

In a simplified way, the total time of a mobile agent's journey across the network consists of the addition of the transfer time on the network, of the time related to the infrastructure, and of the time to run the application. For this application of gathering MIB-II variables, the execution time per network element is the addition of communication intervals between the mobile agent and the translator agent, of the communication between the translator agent and the SNMP agent, of the sending of the GetRequest PDU, of the MIB access, of the receiving of the GetResponse PDU, of the communication between the SNMP agent and the translator agent, and of the communication between the translator agent and the mobile agent.

In the implementation that is only based on the SNMP, we have used the traditional model of this protocol. The manager sends an SNMP packet to an SNMP agent that responds to this manager. Requests are sent to all of the elements to be managed in a sequential manner, i.e., a new request is started after receiving the response from the previous one, until the last network element receives a request and sends the response to the manager. This manager has been implemented directly over the Java Virtual Machine.

We have performed an experimental study in order to evaluate the scalability of the two implementations.

[1] In the Mole platform, mobile agents cannot access resources outside the agent system.

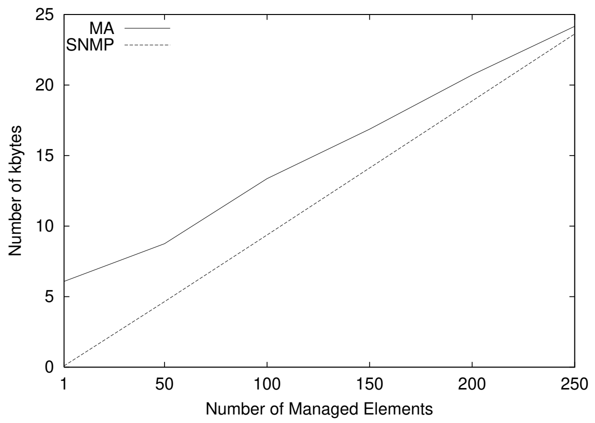


Fig. 2. Number of bytes related to the management station per number of managed elements.

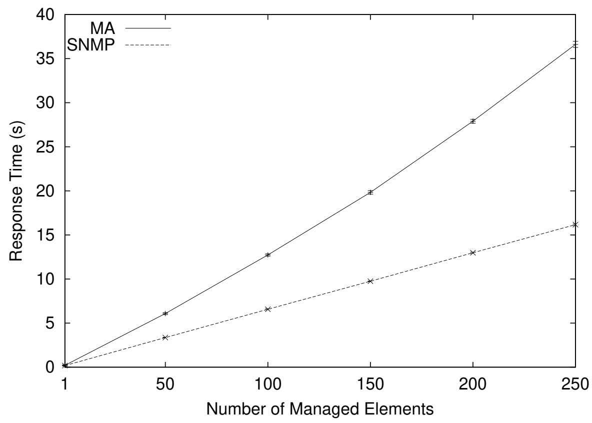


Fig. 3. Response time per number of managed elements.

The topology used in this experiment consists of one management station (host A) and two managed network elements (hosts B and C) interconnected through a 10 Mbps Ethernet LAN. Host A is a Pentium MMX 233 MHz, with 128 Mbytes of memory and Linux Red Hat 6.2. Hosts B and C are Pentium II 350 MHz, respectively with 64 Mbytes and 128 Mbytes of memory, running Linux Red Hat versions 6.1 and 5.2.

In order to evaluate the performance of the two prototypes for a large number of managed elements, we alternately repeat the two hosts B and C, e.g., if we want 5 elements to be managed, we use an itinerary {B, C, B, C, B, and A}. This itinerary information is either in the mobile agent's code or is passed to the manager in the SNMP case.

The considered performance parameters are the number of bytes related to the management station and the response time in retrieving the MIB-II variable *ifInErrors* from the managed elements. This variable denotes the number of received packets discarded because of errors.

The JDK (*Java Development Kit*) 1.1.7 version 3 has been used. In order to limit network performance variations that may cause SNMP packet retransmissions which would alter response time results especially in a loaded network, measurements have been performed on a low loaded network. Both implementations have been tested in the same conditions and using the same itinerary. We have made all the tests with the mobile agent platforms running continuously. The number of managed network elements has been varied from 1 to 250. For each measured parameter, 10 samples have been observed and we have calculated a 99% confidence interval for the mean. If not specified, these intervals are represented in the figures by vertical bars.

The mobile agent carries with itself the name of the variable to be collected, the itinerary, and the already gathered responses. SNMP sends a GetRequest PDU and receives a GetResponse PDU.

The effect of the number of managed elements on the number of bytes related to the management station and on response times has been analyzed. In all figures, we present the obtained mean values.

We are interested in those cases where the management station is attached to bottleneck links and, as a consequence, the traffic that is being sent or received by the management station may overload links and cause losses. Thus, in Fig. 2 we show traffic associated to the management station. It represents the total number of bytes used by the SNMP protocol and the initial and final sizes of the mobile agent plus control messages generated by the Mole platform. We can observe that for few managed elements, SNMP uses fewer bytes than the mobile agent to perform the task (Fig. 2). Nevertheless, as the number of managed elements increases, the total traffic due to several retrievals of GetRequest PDUs exceeds the overhead related to the variable's name, to the itinerary, to the already gathered responses, and to Mole messages. The management station traffic due to the mobile agent utilization becomes less important than SNMP's traffic, as we can conclude by extrapolating the analysis.

According to measurements performed with the *tcpdump* program, the initial size of the mobile agent is approximately 1.5 kbytes. It is also worthy to note that since we have not identified individual samples for the number of bytes, we do not present confidence intervals.

As the required management time is approximately the same for any network elements, SNMP response time grows proportionally with the number of managed elements (Fig. 3). This rule does not extend for the mobile agent. In fact, in the topology used on this experiment, its response time increases more proportionally with the number of managed elements as the agent keeps swelling with more variables being collected on each new network element. Hence, SNMP performs much better than the mobile agent in this topology.

We have also measured the MIBs' access times (times to send/receive GetRequest/GetResponse PDUs) and these access times added to communication times between the mobile agent and translator agents (Fig. 4).

For SNMP and 250 managed elements, 99.6% of the total time is spent on accesses to SNMP MIBs. For the mobile agent and 250 managed elements things work differently as MIB accesses and RPCs take just 52.8% of the total time for the same number of managed elements. For SNMP, accesses to MIBs grow proportionally with the number of managed elements and spend 65 ms per element. These MIB accesses added to RPCs related to the mobile agent and translator agents communications also grow linearly and spend approximately 78 ms per el-

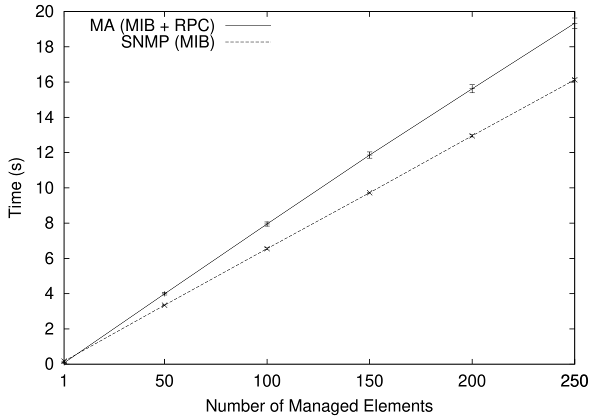


Fig. 4. Time to access the MIBs and for the RPCs.

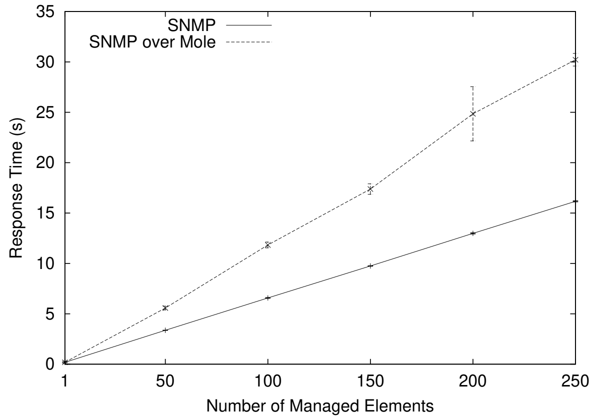


Fig. 6. Response time for SNMP and SNMP over Mole.

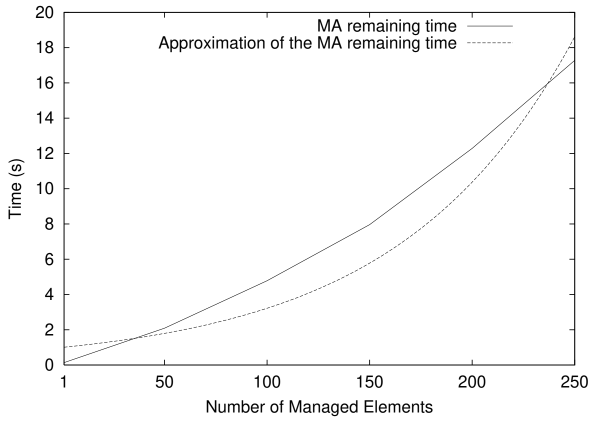


Fig. 5. Mobile agent remaining time.

ement.

The mobile agent remaining time is calculated by the difference between the response time for the mobile agent and the time for accessing MIBs and that of RPCs. This measured remaining time is presented in Fig. 5. Since for this experiment, the agent transmission time is comparably very small in front of the other times that constitute the total response time, the remaining time corresponds to the infrastructure related times, e.g., serialization/deserialization, threads creation, and internal messages transmission. As in Fig. 5, the mobile agent remaining time grows exponentially with the number of managed elements, we have used the measured points to obtain an approximation of this curve as $y = a^x$, where $a = 1.01176$ (the approximation curve is also presented in Fig. 5). This approximation has been chosen to permit simple usage in simulations assessed for more general topologies (Section III).

Mole is a general-purpose infrastructure which is used over the Java Virtual Machine. As such, it may show big processing times for executing Java codes compared to the direct execution over the Java Virtual Machine. In order to verify this platform overhead, even though an agent platform is needless for SNMP, an SNMP manager has also been implemented over the Mole infrastructure (as a stationary agent). According to Fig. 6, the response time increases by approximately 86.9% for 250 managed elements when SNMP is used over Mole. Therefore, the Mole platform has a great influence on the response time. For SNMP over Mole, measurements related to 200 managed elements presented a great confidence interval due to variations in either network performance or computer loads. Thus, a specific mobile agent platform with fewer facilities than the Mole platform may further improve the efficiency gain due to the mobile agent approach.

These measurements have considered the most disadvantageous case for the mobile agent since, in Ethernet, message transmission times are negligible compared with processing times. Measurements have been performed to verify this fact and to obtain parameters closer to the real ones. These will be used in simulations with topologies that are larger and closer to the ones found in the Internet.

III. PERFORMANCE ANALYSIS BY SIMULATION

The applicability of mobile agents technology in carrying out network management tasks is also assessed by simulation.

Network Simulator (NS) [17] is used in these simulations. We have used the functionalities of Ethernet with topologies similar in shape to the Internet and the UDP and TCP protocols. We have written SNMP and mobile agent modules and some UDP and TCP modules have had to be modified in order to allow the transmission of the mobile agents.

NS works with packets sent through a network and usually does not take into account the processing time of the application layer on each end-node. For this reason, some parameters related to network management have been added to the simulation model. These parameters depend on the agent infrastructure, on the operational system, and on computer load. But their use makes simulation results more reliable to a real implementation. Table 1 contains the parameters used in the simulations.

The simulation model assumes that links and nodes have no load and that links are error-free. The Maximum Segmentation Size (MSS) used in the simulations is 1500 bytes. Therefore, there is no fragmentation of SNMP messages since they are already small. For the mobile agent, the initial size is 1500 bytes. After visiting the first element, its size becomes greater than the MSS so it will be fragmented and sent in different packets,

Table 1. Parameters used in the simulations.

Parameter	Value
Initial size of the agent	1500 bytes
Request size for *ifInErrors*	42 bytes
Response size for *ifInErrors*	51 bytes
MIB access time per node for the agent	78 ms
MIB access time per node for the SNMP	65 ms
Related to the agent remaining time	1.01176

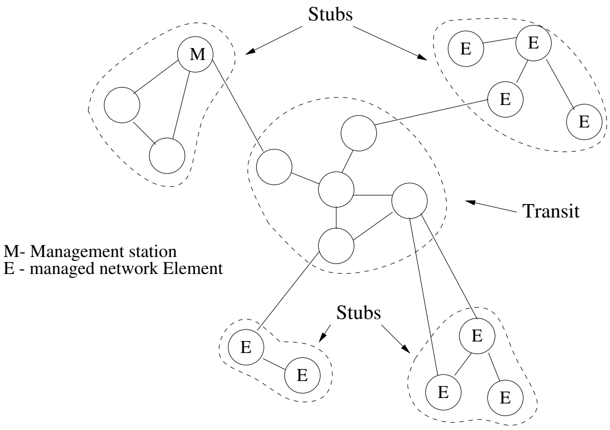


Fig. 7. Network management on a transit-stub topology.

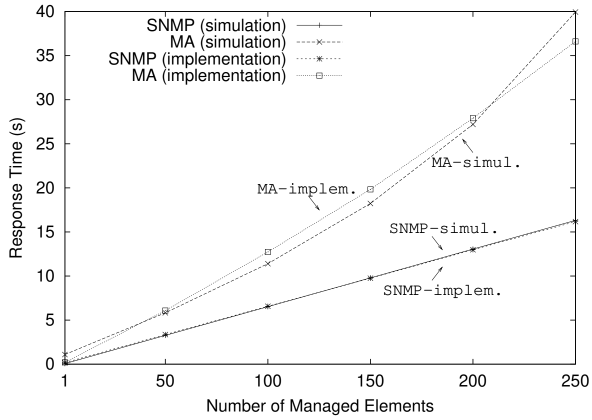


Fig. 8. Response time in implementation and simulation studies.

which incurs performance reductions. Every variable request is sent on a different message. In all simulations, the mobile agent follows a predetermined itinerary. If it is not specified, the mobile agent uses TCP-Reno as the transport protocol because of its widespread use in the Internet. The UDP protocol is used in SNMP simulations. Concerning TCP, the slow-start mechanism is used and the receiver window size is 20, but 30 kbytes are not sent at one time in any of the simulations.

Two kinds of topologies have been used in the simulations. The first one consists of elements in a 10 Mbps Ethernet LAN, with 250 nodes and a latency of 10 μs. The second one is similar in shape to the Internet. This topology is called transit-stub, because each routing domain in the Internet can be classified as either a stub domain or a transit domain. A domain is a stub domain if the path connecting any two nodes u and v goes through that domain and only if either of u or v is in that domain [18]. Transit domains do not have this restriction. The purpose of transit domains is to interconnect stub domains efficiently. A transit domain comprises a set of backbone nodes, which are typically fairly well connected to each other. In a transit domain, each backbone node also connects to a number of stub domains, via gateway nodes in the stubs.

These transit-stub topologies may be used, for example, in the network management of a matrix-branch organization on which a matrix wants to manage the different geographically spread branches. The management strategy used in this experiment for transit-stub topologies considers that the management station belongs to a node of a stub domain and managed network elements are located in other stub domains (Fig. 7). In the matrix-branch case, the management station of the matrix manages the branch routers. Each branch is represented by a stub and contains some routers.

The considered performance parameters are the bandwidth consumption on the management station links and the response time in retrieving the MIB-II variable *ifInErrors*. We have used the LAN topology in order to compare the simulation model to the implementation results of Section II.

Fig. 8 presents the mobile agent and SNMP response times for both of the implementation and simulation studies. We can say that the simulated models reproduce the behavior of the implementations. There is a slight difference in the response time for the mobile agent due to the approximation of the remaining time which has been used in the simulations (Section II).

Mobile agent performance is also evaluated in Internet like topologies in which latencies are much greater than those of LANs. Three different transit-stub topologies created by the topology generator GT-ITM [18] are used. The topologies have 272 nodes and the links of these topologies have a 2 Mbps bandwidth and a latency of a few milliseconds. The management station controls groups of 16 network elements, which is the number of nodes of a stub domain. Management is performed in a predetermined way. First, all elements of a stub are accessed. Then, the next stub is managed until all the 16 stubs are accessed. If not specified, figures present the mean response time or bandwidth consumption for the three topologies.

The response time for SNMP and the mobile agent is analyzed for all of the three different topologies (Fig. 9). The results of the three topologies and a curve linking the mean values are plotted. Fig. 9 shows that the mobile agent's behavior does not change with the topology (the three points coincide). But for SNMP, there is a slight difference in the response time for the three topologies. This variation is due to the great number of SNMP packets that traverse the backbone (transit) links and to the configuration of the backbone nodes that changes with the topology. Fig. 9 also presents mean response times per number of managed elements. For a few set of managed elements, SNMP performs better than the mobile agent because SNMP messages are smaller than the initial size of the mobile agent. As the number of managed elements increases, SNMP response time grows proportionally since the time to manage a stub is approximately the same for all the stubs. For the mobile agent, the response time increases faster when the number of managed

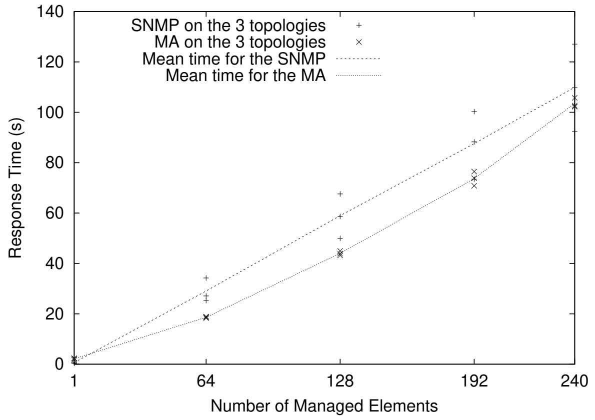


Fig. 9. Response time for the mobile agent and SNMP.

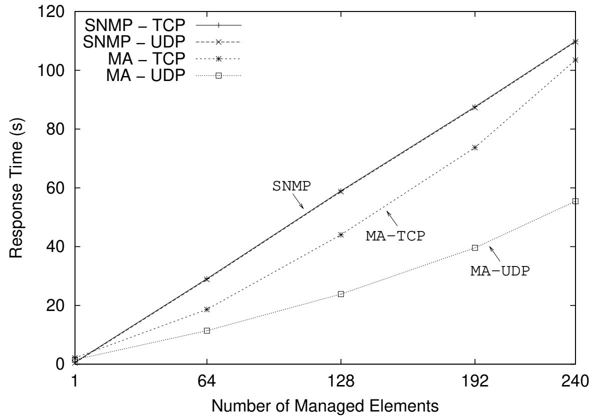


Fig. 11. Response time for TCP and UDP protocols.

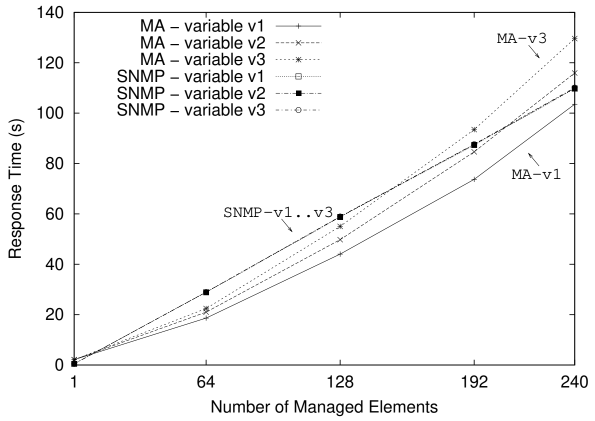


Fig. 10. Response time for different variables.

elements grows due to the incremental size of the mobile agent. By extrapolating the analysis, we can conclude that the mobile agent performs better than SNMP when the number of managed network elements ranges between two limits, an inferior bound and a superior one. The number of messages that pass through a backbone and the size of the mobile agent increasing with network elements collected variables determine these limits.

An analysis related to mobile agent and SNMP's behavior in obtaining different variables is also performed. Three variables are used: variable v_1 is the same variable already used, and variables v_2 and v_3 are *sysORDescr.3* and *sysORDescr.5*. All variables have a request size of 42 bytes and the response sizes are of 51, 87, and 128 bytes. In Fig. 10 and for SNMP, the obtained variable does not have a great influence because the number of bytes exchanged between the management station and the managed elements is small. For the mobile agent, the response time increases as the number of exchanged bytes raises (e.g., from variable v_1 to variable v_3). In this experiment, when the number of managed elements is 240, the response time for the mobile agent increases by 25.2%, obtaining variable v_3 instead of v_1.

The Mole platform uses the reliable connection-oriented TCP protocol, while SNMP uses the connectionless UDP protocol. In order to verify the influence of the transport protocol to transfer the data (not considering the connection phase of the TCP protocol which is not provided by the NS simulator), we have performed simulations with the mobile agent over UDP and SNMP over TCP-Reno. In this experiment and for SNMP, the protocol does not have a great influence on the response time. But for the mobile agent, this time decreases significantly when UDP is used (Fig. 11). This is due to the ACK transmission and mainly to TCP congestion control, i.e., the *slow-start* mechanism. Because of the use of this mechanism, the segmented packets of the mobile agent are not sent at one time which directly increases the response time. This mechanism does not influence SNMP because of the its small packet size which is never greater than congestion or receiver's windows sizes.

It has been shown that the mobile agent size increases with the number of visited nodes. As a consequence, migration becomes difficult. We also evaluate the performance gain of a strategy in which the mobile agent returns to the management station to reduce its size. This experiment considers that in one "trip" the mobile agent visits a fixed number of nodes. Then, it returns to the management station to "unload" collected data finishing this trip. After that, the agent restarts the task of gathering variables on the remaining nodes. The considered variables are the three already presented, but now we aim at retrieving variables from all of the 240 managed elements. The number of visited nodes per trip varies from 1 to 240.

Fig. 12 shows that for a few managed elements per trip, the response time decreases sharply when the number of managed elements per trip increases. This is because in one trip the agent visits few nodes and returns to the management station passing through the backbone links. As the number of visited nodes per trip increases, the response time keeps on decreasing until a specific point. Then, it starts to increase due to the agent migration difficulty related to the agent size. The curve presents a sawtooth format because when the number of elements to be visited in a trip is not a multiple of the number of elements of a stub (16 in this experiment), the mobile agent traverses several stubs during a trip. Consequently, it passes more times through the transit nodes and hence increases the response time. The "optimum point" for this experiment varies with the size of the mobile agent and consequently with the variable to be obtained. For example, for variable v_1, visiting 48 nodes and returning to the management station provides the best result. Response time

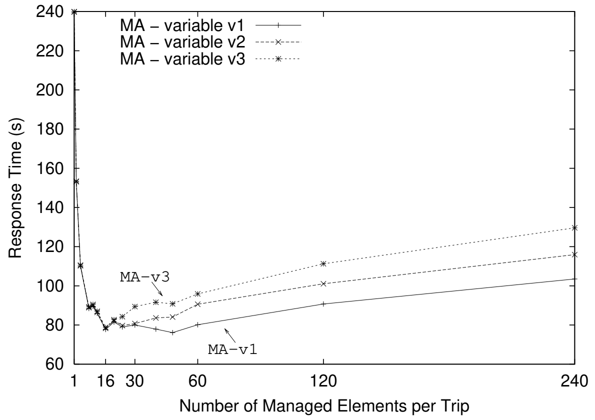


Fig. 12. Response time when returning to the management station.

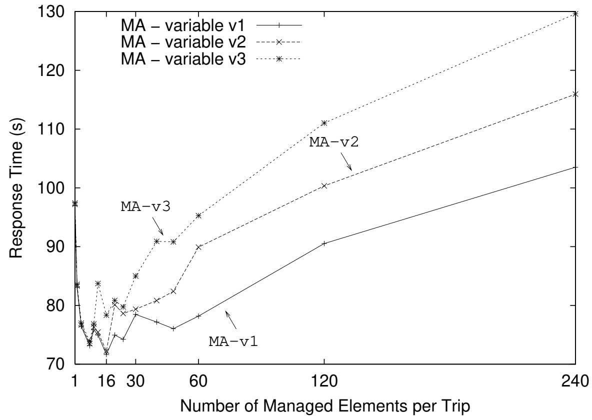


Fig. 14. Response time when sending data to the management station.

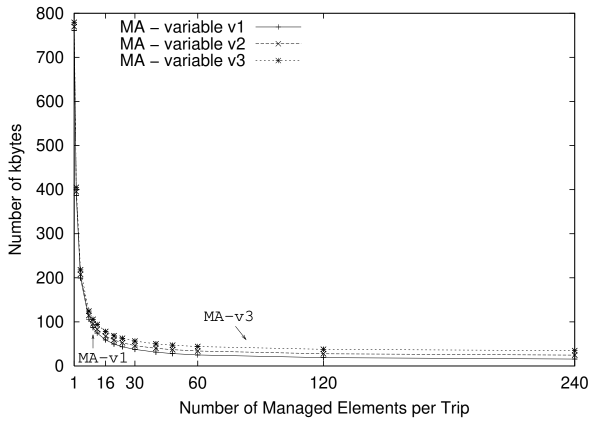


Fig. 13. Bandwidth consumption on the management station links when returning.

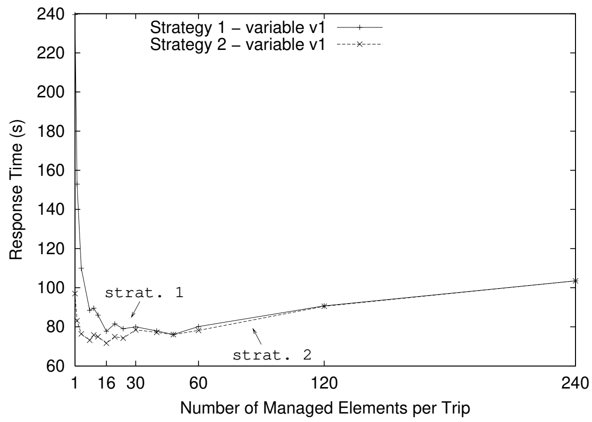


Fig. 15. Response time for the two strategies.

decreases by 27%, 32%, and 39%, for variables v_1, v_2, and v_3, when comparing with obtaining all 240 variables in one trip, i.e., the traditional mobile agent use.

For bandwidth consumption on the management station links, it decreases when the number of managed elements per trip increases, as in Fig. 13. This is because the number of transmitted and received bytes by this station is larger when there are more mobile agent returns to the management station.

The strategy of sending the results to the management station instead of returning to this station is also analyzed. In Fig. 14, we can see the same sawtooth effect. The response time decreases by 31%, 38%, and, 42%, for variables v_1, v_2, and v_3, comparing each optimum point with obtaining the 240 variables in one trip.

In Fig. 15, we see that the strategy of sending data performs better than the one of returning to the management station when the number of per trip managed elements is small, since the mobile agent's code is not sent to the management station and the mobile agent only returns to the management station after finishing its task. As this number of elements per trip increases, both approaches behave almost the same because the size of the agent gathered data is much greater than the size of the mobile agent's code and because of the smaller number of returns to the management station.

Fig. 16 shows that for a few per trip managed elements, the bandwidth consumption on the management station links decreases sharply when the number of managed elements increases, because in one trip the agent visits a small number of nodes and sends data to the management station. In this first time, the number of acknowledgments sent to the management station decreases when the number of per trip managed elements increases. As the number of per trip visited nodes raises, bandwidth utilization continues to decrease until reaching a point where it tends to a specific value. This because of the enlarging agent size, which generates more fragmentation and hence much more acknowledgments.

IV. CONCLUSION

We have compared two prototype implementations for gathering MIB-II (Management Information Base - II) variables on managed elements: A mobile agent-based one and an only SNMP based one. Parameters obtained from the implemented prototypes have been used in simulations on large Internet-like topologies. Both of the implemented prototypes have used the SNMP protocol (the AdventNet SNMP library and the ucd-snmp package) to gather MIB-II variables and have been tested

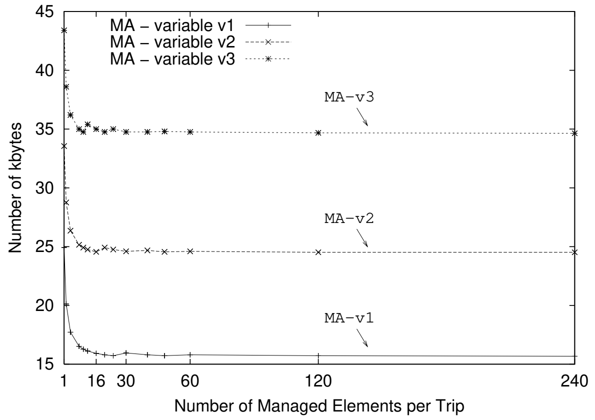


Fig. 16. Bandwidth consumption on the management station links when sending data.

on an Ethernet LAN.

Results show that SNMP uses a larger number of management station related bytes when the number of managed elements exceeds a value related to the overhead of several retrievals of GetRequest PDUs and that the mobile agents require a higher processing capacity. The mobile agent infrastructure turns the execution of Java code slower. We have compared SNMP response time directly over the Java Virtual Machine with one for the SNMP over the Mole infrastructure. The response time increases by approximately 86.9% for 250 managed elements when SNMP is used over Mole. The topology used for the measurements is adverse to the mobile agent, since the great availability of bandwidth on the Ethernet turns negligible the message transmission times compared with processing times. For this topology, the mobile agent becomes very sensitive to the processing capacity of the machines. We have conducted some experiments substituting the managed hosts by a Pentium MMX 233 MHz, with 64 Mbytes of memory and Linux Red Hat version 5.2, and by a Pentium 133 MHz with 32 Mbytes of memory and Linux Red Hat version 6.1. These experiments have shown that there was no performance degradation when using SNMP but the response time has grown up to 89.5% for the mobile agent compared with the results with the higher processing capacity of the machines. For this topology, SNMP performs better than the mobile agent.

Simulations of the two implementations have also been performed to obtain results for large Internet-like topologies. For transit-stub topologies, the response time experiences a significant decrease when UDP is used since TCP slow-start mechanism does not send all of the mobile agent related packets at once.

The mobile agent outperforms SNMP when the number of managed elements ranges between two limits, an inferior bound and a superior one, determined by the number of messages that pass through the backbone links and by the mobile agent size that grows with the variables collected on network elements. Moreover, better management performance is experienced when the mobile agent returns to the management station after visiting a fixed number of nodes, since this allows the mobile agent size to be limited. The response time decreases down to 39% for the variables that have been used. Better results are obtained when the agent only sends data to the management station. In this case, the improvement is up to 42%.

We conclude that the mobile agent paradigm significantly improves network management performance when subnetworks are managed remotely, especially if the links between the management station and the elements to be managed are of high-cost (have a small bandwidth and a large latency).

This work opens the road for some future novel research directions. In order to improve the mobile agent performance, a direct MIB access could be implemented. This helps fusing the SNMP and translator agents into one. Other mobile agent platforms could also be tested. The use of intelligence over the mobile agents to perform a reactive management with task delegation and verification of when certain thresholds are exceeded could equally be studied. SNMPv2 could be compared to mobile agents for transit-stub topologies. Proxy managers would be used on a node of a stub in order to avoid large latencies between the manager and the agents for SNMPv1. Other approaches that can face the latency problem are the multi-threaded version of the SNMP manager and the multi-agent solution. The response time can be improved by employing simultaneous requests that are sent in the multi-threaded SNMP approach as well as through the use of multiple agents which can manage several stubs at the same time in the multi-agent solution.

ACKNOWLEDGMENTS

This work has been supported by UFRJ, CNPq, CAPES, COFECUB, and FAPERJ.

REFERENCES

[1] M. Baldi and G. P. Picco, "Evaluating the tradeoffs of mobile code design paradigms in network management applications," in *Proc. 20th ICSE'98*, (Kyoto, Japan), Apr. 1998m pp. 146–155.

[2] D. Gürer, V. Lakshminarayan, and A. Sastry, "An intelligent-agent-based architecture for the management of heterogeneous networks," in *Proc. 9th IFIP/IEEE DSOM'98*, (Delaware, USA), Oct. 1998.

[3] V. A. Pham and A. Karmouch, "Mobile software agents: An overview," *IEEE Commun. Mag.*, vol. 36, pp. 26–37, July 1998.

[4] A. Bieszczad, B. Pagurek, and T. White, "Mobile agents for network management," *IEEE Commun. Surveys*, vol. 1, Sept. 1998.

[5] M. Zapf, K. Herrmann, and K. Geihs, "Decentralized SNMP management with mobile agents," in *Proc. 6th IFIP/IEEE IM'99*, (Boston, USA), May 1999, pp. 623–635.

[6] M. El-Darieby and A. Bieszczad, "Intelligent mobile agents: Towards network fault management automation," in *Proc. 6th IFIP/IEEE IM'99*, (Boston, USA), May 1999, pp. 610–622.

[7] A. Outtagarts, M. Kadoch, and S. Soulhi, "Client-server and mobile agent: Performances comparative study in the management of MIBs," in *Proc. MATA'99*, (Ottawa, Canada), Oct. 1999, pp. 69–81.

[8] G. P. Picco, "Mobile agents: An introduction," *Microprocessors and Microsystems*, vol. 25, pp. 65–74, Apr. 2001. Elsevier Science.

[9] A. Puliafito, S. Riccobene, and M. Scarpa, "An analytical comparison of the client-server, remote evaluation and mobile agent paradigms," in *Proc. ASA/MA'99*, (California, USA), Oct. 1999.

[10] A. Liotta, G. Knight, and G. Pavlou, "On the performance and scalability of decentralised monitoring using mobile agents," in *Proc. 10th IFIP/IEEE DSOM'99*, (Zurich, Switzerland), Oct. 1909, pp. 3–18.

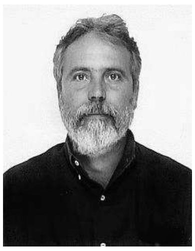
[11] C. Bohoris, G. Pavlou, and H. Cruickshank, "Using mobile agents for network performance management," in *Proc. IEEE/IFIP NOMS'00*, (Honolulu, Hawaii), Apr. 2000, pp. 637–652.

[12] D. Gavalas, *et al.*, "Using mobile agents for distributed network performance management," in *Proc. 3rd IATA'99*, (Stockholm, Sweden), Aug. 1999.

[13] A. Sahai and C. Morin, "Towards distributed and dynamic network management," in *Proc. IEEE/IFIP NOMS'98*, (New Orleans, USA), Feb. 1998.

[14] J. Baumann, *et al.*, “Mole - concepts of a mobile agent system,” *World Wide Web*, vol. 1, no. 3, pp. 123–137, 1998.
[15] Advent Network Management Inc., “AdventNet SNMP release 2.0.” http://www.adventnet.com, 1998.
[16] “UCD-SNMP project.” http://ucd-snmp.ucdavis.edu/, 2000.
[17] K. Fall and K. Varadhan, “NS Notes and Documentation,” tech. rep., The VINT Project, Jan. 1999.
[18] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, “A quantitative comparison of graph-based models for internet topology,” *IEEE/ACM Trans. Networking*, vol. 5, pp. 770–783, Dec. 1997.

Marcelo Gonçalves Rubinstein was born in Rio de Janeiro, Brazil, on August 29, 1971. He received the B.Sc. degree in Electronics Engineering and the M.Sc. and the D.Sc. degrees in Electrical Engineering from the Federal University of Rio de Janeiro, Brazil, in 1994, 1996, and 2001 respectively. From January to September 2000, he was at the PRiSM Lab from the University of Versailles, France. From December 2001 to February 2002, he was a research fellow at the Grupo de Teleinformática e Automação from the Federal University of Rio de Janeiro. He is Associate Professor at the State University of Rio de Janeiro. His major interests are mobile agents, network management, multimedia applications, and mobile networks.



Otto Carlos M. B. Duarte was born in Rio de Janeiro, Brazil, on October 23, 1953. He received the Electronic Engineer degree and the M.Sc. degree in Electrical Engineering from the Federal University of Rio de Janeiro, Brazil, in 1976 and 1981, respectively, and the Dr.Ing. degree from ENST/Paris, France, in 1985. He is Professor at the Federal University of Rio de Janeiro. From January 1992 to June 1993 he has worked at MASI Laboratory in the University of Paris 6. In 1995, he has spent three months at International Computer Science Institute (ICSI) associated to the University of California at Berkeley. Presently, he is heading the computer network group (Grupo de Teleinformática e Automação- GTA). His major research interests are in high speed communications, mobility, security and QoS guarantees.

Guy Pujolle received his Ph.D. and Thèse d’Etat degrees in computer science from the University of Paris IX and XI in 1975 and 1978, respectively. He is currently professor at the University of Paris VI. He is chair of IFIP Working Group 6.2 on Network and Internetwork Architectures and is a governor of ICCC. He has published widely in the areas of computer systems modeling and performance, queuing theory, and high-speed networks. His research interests include the analysis and modeling of data communication systems, protocols, high-performance networking, intelligence in networking, and wireless networks. He has been Professor Honoris Causa of Beijing University since 1988. He was awarded the Special Seymour Cray Award in 1991 for his research and the Silver Core from IFIP in 1995.