Evaluating the Performance of a Network Management Application Based on Mobile Agents

Marcelo G. Rubinstein[1], Otto Carlos M. B. Duarte[2], and Guy Pujolle[3]

[1] Depto. de Eng. Eletrônica e Telecom., Universidade Estadual do Rio de Janeiro, FEN, Rua São Francisco Xavier, 524, 20550-013, Rio de Janeiro RJ, Brazil,
[2] Grupo de Teleinformática e Automação, Universidade Federal do Rio de Janeiro, COPPE/EE, CP 68504, 21945-970, Rio de Janeiro RJ, Brazil,
[3] Laboratoire LIP6-CNRS, Université Pierre et Marie Curie, 4, Place Jussieu, 75252, Paris Cedex 05, France

Abstract. This paper analyzes mobile agent performance in network management, comparing it with the client-server model used by the SNMP (Simple Network Management Protocol). Response time results show that the mobile agent performs better than the SNMP when the number of managed elements ranges between two limits determined by the number of messages that pass through a backbone and by the mobile agent size that grows with the variables collected on the network elements.

Keywords

Mobile agents, network management, and scalability

1 Introduction

Most network management systems use SNMP (Simple Network Management Protocol) [1] and CMIP (Common Management Information Protocol) [2] protocols, which are based on a centralized paradigm. These protocols use the client-server model, on which the management station acts as a client that provides a user interface to the network manager and interacts with agents, which are servers that manage remote access to local information stored in a Management Information Base (MIB).

Performance management is one of the management functional areas identified in OSI Systems Management and addresses the availability of management information, in order to be able to determine the network load [3]. This kind of management needs access to a large quantity of dynamic network information, which is collected by periodic polling.

The operations available to the management station for obtaining access to the MIB are very low-level. This fine grained client-server interaction, called micro-management, and the periodic polling generate an intense traffic that overloads the management station [4], resulting in scalability problems. Network management can be distributed and scaled by the use of mobile agents, which are programs that help users to perform tasks on the network, acting on behalf of these users. These agents move to the place where data are stored and select information the user wants; saving bandwidth, time, and money.

This paper analyzes the performance of mobile agents in network management, which is also being investigated by several researchers. Baldi et al. [4] evaluate the tradeoffs of mobile code design paradigms in network management applications by developing a quantitative model that provides the bandwidth used by traditional and mobile code design of management functionalities. Bohoris et al. [3] present a performance comparison between mobile agents, CORBA, and Java-RMI on the management of an ATM switch running an SNMP agent. Response time and bandwidth utilization results are presented for the transfer of an array of objects (fictitious data). Gavalas et al. present experimental implementation results for the transfer of an aggregation of multiple variables on a local network of a few nodes. They also describe applications that use mobile agents to acquire atomic snapshots of SNMP tables and to get objects, from SNMP tables, that meet specific criteria [5]. Sahai and Morin [6] perform measurements of bandwidth utilization of mobile agent and client-server applications on an Ethernet LAN of a few nodes. None of these papers concerns the problem of scalability of network management based on mobile agents on a complex network with a high number of nodes and similar in shape to the Internet. In this paper, we compare the scalability of the network management based on mobile agents against traditional SNMP through the analysis of simulation and implementation results. Two prototypes of an application that gathers MIB-II variables, one based on mobile agents and the other only based on the SNMP, have been created and tested on a LAN. By acquiring parameters related to the network management and to the agent infrastructure, new results are obtained on large topologies similar in shape to the Internet.

This paper is organized as follows. Section 2 describes main network management systems used nowadays. Section 3 presents the implemented prototypes and measurement results. Section 4 reports simulation results. At last, concluding remarks are presented in Section 5.

2 Network Management Systems

In the SNMP, operations available to the management station for accessing the MIB are very low-level. This interaction does not scale well because of the generation of intense traffic and computational overload on the management station [4].

Some steps towards decentralization have already been taken by IETF and ISO organizations. In event notification, SNMP 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. ISO uses more complex agents that have higher processing capacity. In both the approaches, the agent is only responsible for the notification of an event.

A more decentralized approach is adopted in SNMPv2 [1] (SNMP version 2), on which there are multiple top-level management stations, called management servers. Each such server is responsible for managing agents, but it can delegate responsibility to an intermediate manager. This manager, also called proxy agent, plays the role of a manager in order to monitor and control the agents under its responsibility and also works as an agent to provide information and to accept control from a higher-level management server. Version 3 of the SNMP, SNMPv3 [1], incorporates a new security scheme to be used with SNMPv2 (preferred) or SNMPv1. SNMPv3 is not a stand-alone replacement for SNMPv1 or SNMPv2.

The RMON (Remote MONitoring) [7] 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 among 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 the traffic around the management station, but as the computational power of the network nodes is increasing, it is possible to delegate more complex management functions to nodes. Moreover, in order to satisfy the diverse needs of today's network, new network management systems that analyze data, take decisions, and take proactive measures to maintain the Quality of Service (QoS) of the network must be developed. Mobile agents seem to be a good alternative to satisfy these needs.

Main advantages that may justify mobile agent utilization in network management are: reduced cost by using a semantic compression, which filters and selects only relevant information; asynchronous processing that allows the decoupling from the home node; flexibility that permits the substitution of the behavior for management agents in real-time; and autonomy, since the agent can take decisions, performing a reactive management based on task delegation.

Since SNMPv2 is not as spread as SNMPv1 and network management based on SNMPv1 does not scale when size or complexity of the network increases, mobile agents can be used to increase network management scalability.

3 Implementation of a Management Application

We compare two different solutions for gathering MIB-II [8] variables on managed elements: a mobile agent-based one and one only based on the SNMP.

The Mole infrastructure [9] is used in the mobile agent implementation. This system provides the functionality for the agents to move, to communicate with each other, and to interact with the underlying computer system. Two different kinds of agents are provided: system agents and user agents. System agents are usually interface components to resources outside agent systems. They have more rights than non-system agents (e.g., only system agents can read or write to a file), but they are not able to migrate. User agents are agents that have a "foreigner status" at a location, which means that they are not allowed to do something outside the agent system as long as they can not convince a system agent to give them access to outside resources [9].

Mole uses TCP to transfer mobile agents, which are implemented in Java. A weak migration scheme is provided, where only the state related to data, which contains global and instantiated variables, is transfered. As a consequence, the programmer is responsible for encoding the agent's execution state, which includes local variables, parameters, and execution threads, in program variables. This migration scheme is implemented by using the object serialization of Java. After the calling of the migrateTo()-method by an agent thread, all threads belonging to the agent are suspended. The agent is serialized by creating a system-independent representation. This serialized version is sent to the target that reinstantiates the agent. A new thread is started and as soon as this thread assumes control of the agent, a message is sent to the source that finishes all threads belonging to the agent and removes it from the system.

Both the implemented prototypes, one with mobile agents and the other without, use the SNMP protocol to gather MIB-II variables. The AdventNet SNMP library [10] and the snmpd from package ucd-snmp are used on the prototypes. The AdventNet SNMP package contains APIs to facilitate the implementation of solutions and products for network management. Version 2.2 of the AdventNet SNMPv1 has been used. The daemon snmpd, which is included in the Linux Red Hat, is an SNMP agent that responds to SNMP request packets. The package versions used on this experiment have been the 3.5.3 (for machines running the Red Hat 5.2) and the 4.0.1 (for the Red Hat 6.x).

3.1 The Two Implemented Prototypes

The mobile agent implementation (Figure 1) consists of one mobile agent, which migrates to all network elements to be managed, one SNMP agent, which accesses the MIB-II variables, and one translator agent, which converts the mobile agent request into an SNMP request. The mobile agent migrates to a network element (arc 1 of Figure 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) that is passed to the mobile agent (arc 5). Then, the mobile agent goes to the next element (arc 6) and restarts its execution. After finishing its task, which consists of visiting all network elements to be managed, the mobile agent returns to the management station (arc n).

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. The manager sends requests to all elements to be managed, one after the other, 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 in Java directly over the Java Virtual Machine.

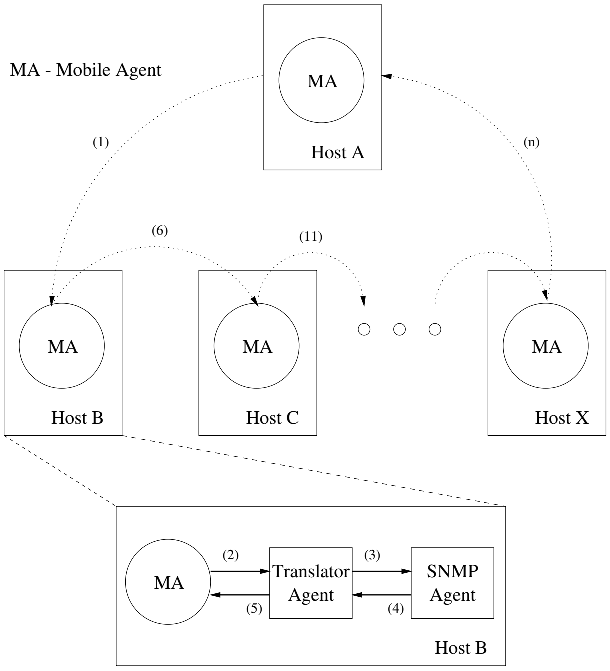

Fig. 1. Network management by using a mobile agent.

3.2 Experimental Study

We perform an experimental study in order to evaluate the scalability of the two implementations. The topology used on 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 running Linux Red Hat 6.2. Hosts B and C are Pentiuns II 350 MHz, respectively with 64 Mbytes and 128 Mbytes of memory and running Linux Red Hat versions 6.1 and 5.2.

In order to evaluate the performance of the two prototypes for a great number of managed elements, we alternately repeat the two elements B and C, e.g., if we want 5 elements to be managed, we use an itinerary {B, C, B, C, B, and A}.

The considered performance parameter is response time in retrieving the MIB-II [8] variable *ifInErrors* from elements to be managed. 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. All measurements have been performed early in the morning or at night in order to limit the variations of network performance, which would influence the response time results. Both the implementations have been tested in the same conditions, using the same itinerary. We have made all the tests with the mobile agent platforms running uninterruptedly. 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 mean. 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 gotten responses. The SNMP sends a GetRequest PDU and receives a GetResponse PDU.

The effect of the number of managed elements in response time has been analyzed. In all figures, we present the sample mean.

Response time for the SNMP grows proportionally with the number of managed elements, since the time to manage a network element is approximately the same for all network elements (Figure 2). For the mobile agent, response time increases faster when the number of managed elements grows, due to the mobile agent size that grows with the collected variables on each network element. In the topology used on this experiment, the SNMP performs much better than the mobile agent.

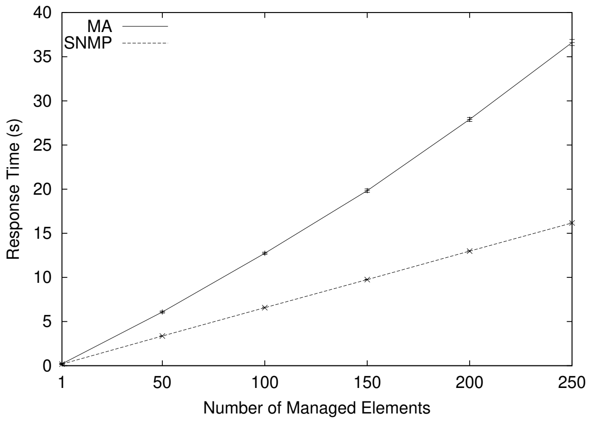

Fig. 2. Response time per number of managed elements.

Figure 3 presents time to access the MIBs and for the RPCs related to the communication between the mobile agent and translator agents. In this experiment and for the SNMP, for 250 managed elements, 99.6 % of the total time is spent on access to the MIBs. For the mobile agent, the access to the MIBs and the RPCs take 52.8 % of the total time for 250 managed elements. For the SNMP, the access to the MIBs grows proportionally with the number of managed elements and spends 65 ms per element. This MIBs access added to the RPCs related to the communication between the mobile agent and translator agents also grow linearly and spend approximately 78 ms per element.

The mobile agent remaining time is calculated by the difference between the total time for the mobile agent and the time for accessing the MIBs and

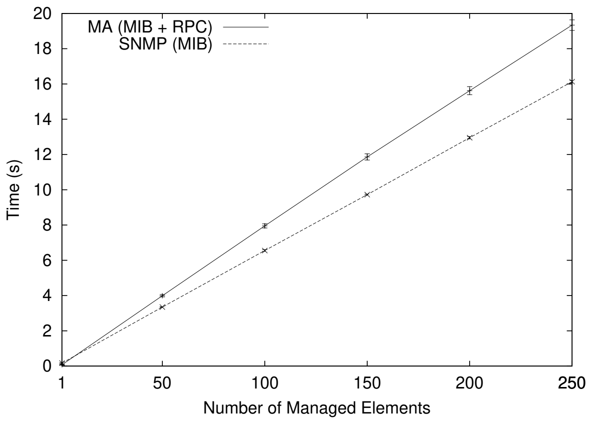

Fig. 3. Time to access the MIBs and for the RPCs.

for the RPCs (Figure 4). Since, for this experiment, agent transmission time is very small comparing to other times that constitute the total response time, the remaining time corresponds to infrastructure related times, e.g., serialization/deserialization, threads creation, and internal messages transmission.

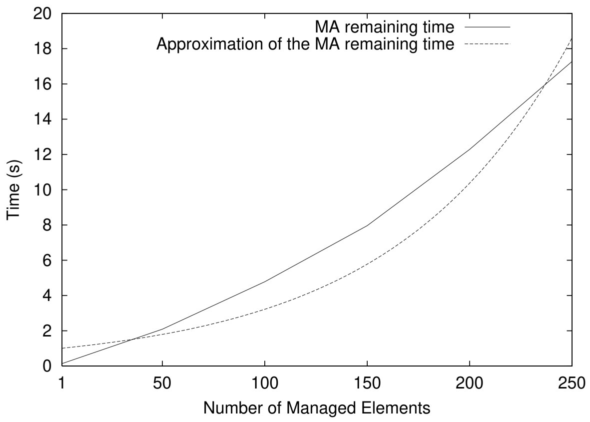

Fig. 4. Mobile agent remaining time.

The mobile agent remaining time grows exponentially with the number of managed elements, so the curve of the Figure 4 can be approximated to:

$$y = a^x, \text{ where } a = 1.01176 \ . \tag{1}$$

This approximation has been chosen to allow, in a simple way, its use in simulations assessed for more general topologies (Section 4).

4 Performance Analysis by Simulation

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

The Network Simulator (NS) [11] is used in these simulations. This discrete-event simulator provides several implemented protocols and mechanisms to simulate computer networks with node and link abstractions. In these simulations, we have used the functionalities of Ethernet, topologies similar in shape to the Internet, and UDP and TCP protocols. Some UDP and TCP modules of the NS have had to be modified in order to allow the transmission of mobile agents.

The NS works with packets sent through a network and usually does not take into account processing time of the application layer on each 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 turns simulation results more reliable to a real implementation. Table 1 contains the parameters used in the simulations.

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 remaining time for the agent	1.01176

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 small. For the mobile agent, since the initial size is 1500 bytes, after visiting the first element, its size will be higher than the MSS, and so the agent will be fragmented and sent in different packets, damaging the performance. Every request of a variable is sent on a different message. In all simulations, the mobile agent follows a predetermined itinerary. The mobile agent uses the TCP-Reno as a transport protocol, because of its great use in the Internet, and the UDP protocol is used in the SNMP simulations.

Two kinds of topologies have been used in the simulations. The first type consists of elements in a 10 Mbps Ethernet LAN, with 250 nodes and latency of 10 μs. The second kind 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 [12]. A domain is a stub domain if the path connecting any two nodes u and v goes through that domain only if either u or v is in that domain. 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 can be used in the network management of a matrix-branch organization, on which a matrix wants to manage their branches spread geographically. 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 (Figure 5). In the matrix-branch case, the management station from the matrix manages the branch routers and each branch is represented by a stub and contains some routers.

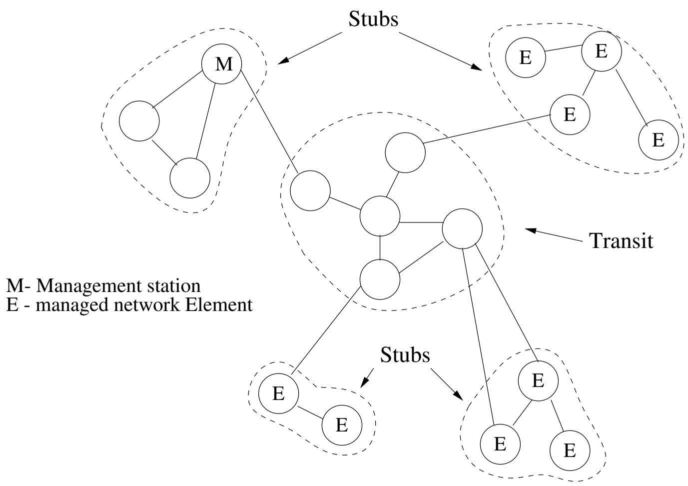

Fig. 5. Network management on a transit-stub topology.

The considered performance parameter is response time in retrieving the MIB-II variable *ifInErrors*.

We have used the LAN topology in order to compare the simulation model with the implementation results of Section 3.2.

Figure 6 presents response time for the mobile agent and for the SNMP, in implementation and simulation studies. We can say that the simulated models reproduce the behavior of the implementations. There is a little difference in the response time for the mobile agent due to the approximation of the remaining time that has been used in the simulations (Section 3.2).

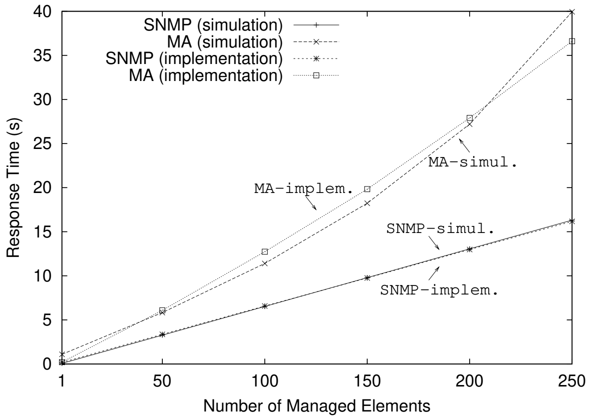

Fig. 6. Response time for implementation and simulation studies.

Mobile agent performance is also evaluated in a situation closer to the one found on the Internet, on which latencies are much greater than on LANs. Three different transit-stub topologies created by the topology generator GT-ITM [12] are used. The topologies have 272 nodes and links of these topologies have a 2 Mbps bandwidth and 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: all elements of a stub are accessed, after that, the next stub is managed, until all the 16 stubs are accessed. If not specified, figures present mean response time for the three topologies.

Figure 7 shows that the mobile agent's behavior does not change with the topology, but for the SNMP, there is a little difference in 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. Figure 7 also presents mean response time. For a small number of managed elements, the SNMP performs better than the mobile agent, due to the fact that the SNMP messages are smaller than the initial size of the mobile agent. As the number of managed elements increases, response time for the SNMP grows proportionally, since the time to manage a stub is approximately the same for all stubs. For the mobile agent, response time increases faster when the number of managed 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 the SNMP when the number of managed network elements ranges between two limits, an inferior and a superior one, respectively determined by the number of messages that pass through a backbone and the size of mobile agent that grows with the variables collected on network elements.

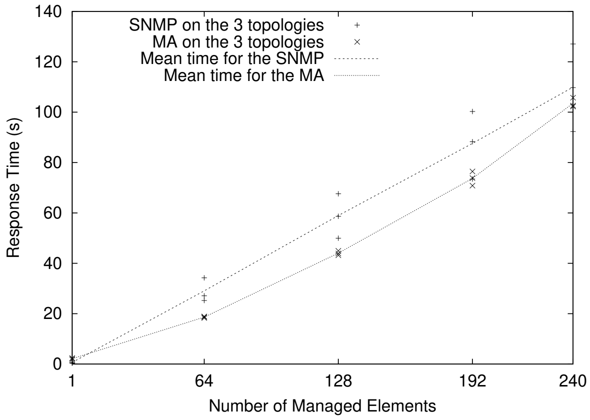

Fig. 7. Response time for the mobile agent and for the SNMP.

5 Conclusion

This work has analyzed the scalability of mobile agents in network management. The performance of mobile agents has been compared with the SNMP (Simple Network Management Protocol) one.

We have compared two prototype implementations for gathering MIB-II (Management Information Base - II) variables on managed elements: a mobile agent-based one and the pure SNMP. Results show that the mobile agents require a higher processing capacity and that the SNMP uses a larger number of messages related to the management station when the number of managed elements exceeds a value related to the overhead of several retrievals of GetRequest PDUs. The mobile agent infrastructure turns the execution of Java code slower, mainly because of serialization/deserialization, threads creation, and internal messages transmission. The topology used on the measurements is adverse to the mobile agent, since the great availability of bandwidth on the Ethernet turns message transmission times negligible comparing with processing times. Therefore, in this topology, the SNMP performs much better than the mobile agent.

Simulations of the two implementations have also been performed in the NS Network Simulator, in order to obtain results on large topologies similar in shape to the Internet. Response time results show that the mobile agent performs better than the SNMP when the number of managed elements ranges between two limits, an inferior and a superior one, respectively determined by the number of messages that pass through a backbone and by the mobile agent size that grows with the variables collected on network elements.

In a general way, we conclude that the mobile agent paradigm significantly improves the network management performance when subnetworks must be managed remotely; mainly if the links between the management station and the elements to be managed have a small bandwidth and a large latency.

Acknowledgements

This work has been supported by UFRJ, FUJB, CNPq, CAPES, COFECUB, and REENGE. We would like to thank FAPERJ for the grant to Mr. Rubinstein during the execution of this work on the Departamento de Engenharia Eletrônica e de Computação da UFRJ.

References

1. Stallings, W.: SNMP and SNMPv2: The infrastructure for network management. IEEE Communications Magazine **36** (1998) 37–43
2. Yemini, Y.: The OSI network management model. IEEE Communications Magazine **31** (1993) 20–29
3. Bohoris, C., Pavlou, G., Cruickshank, H.: Using mobile agents for network performance management. In: IEEE/IFIP Network Operations and Management Symposium (NOMS'00), Honolulu, Hawaii (2000) 637–652
4. Baldi, M., Picco, G.P.: Evaluating the tradeoffs of mobile code design paradigms in network management applications. In: 20th International Conference on Software Engineering (ICSE'98), Kyoto, Japan (1998) 146–155
5. Gavalas, D., Greenwood, D., Ghanbari, M., O'Mahony, M.: Mobile software agents for decentralised network and systems management. Microprocessors and Microsystems **25** (2001) 101–109
6. Sahai, A., Morin, C.: Towards distributed and dynamic network management. In: IEEE/IFIP Network Operations and Management Symposium (NOMS'98), New Orleans, USA (1998)
7. Waldbusser, S.: Remote network monitoring management information base. RFC 1757 (1995)
8. McCloghrie, K., Rose, M.: Management information base for network management of TCP/IP-based internets: MIB-II. RFC 1213 (1991)
9. Baumann, J., Hohl, F., Straber, M., Rothermel, K.: Mole - concepts of a mobile agent system. World Wide Web **1** (1998) 123–137
10. Advent Network Management Inc.: AdventNet SNMP release 2.0. http://www.adventnet.com (1998)
11. Fall, K., Varadhan, K.: NS Notes and Documentation. Technical report, The VINT Project (1999)
12. Zegura, E.W., Calvert, K.L., Donahoo, M.J.: A quantitative comparison of graph-based models for internet topology. IEEE/ACM Transactions on Networking **5** (1997) 770–783