A Hybrid approach to Network Performance Monitoring based on Mobile Agents and CORBA

Christos Bohoris, George Pavlou, Antonio Liotta

Centre for Communication Systems Research, School of Electronics, Computing and Mathematics, University of Surrey, Guildford, Surrey, GU2 7XH, UK
{C.Bohoris, G.Pavlou, A.Liotta}@eim.surrey.ac.uk

Abstract. While mobile agent frameworks were initially thought as rivals to static distributed objects the two approaches should ideally coexist. Mobile agents can allow the easy programmability of managed nodes, while static distributed objects typically offer better performance. Real synergy could be achieved in a network management system if entities with a stationary role could be provided using static objects that collaborate with mobile agents in a system that combines the best of both frameworks. In this paper we present such a hybrid approach, illustrated through a prototype network performance monitoring system based on an integration of Mobile Agents and CORBA. Our assessment of this system based on our practical experiences and experimental evaluation shows that such a hybrid approach can provide an easy and favorable solution for enhancing existing CORBA-based network management systems.
Keywords. Mobile Agents, CORBA, Performance Monitoring, Distributed Objects.

1 Introduction and Background

In recent years we have witnessed an evolution in network management approaches driven by the need of addressing the requirements of modern networks becoming increasingly large, sophisticated and complex. The initial protocol-based network management solutions, proposed in the early 90's as exemplified by the widely used Simple Network Management Protocol (SNMP) [1], are today particularly limiting due to their centralized and static nature. The approach is centralized as it relies on a limited set of capabilities at network nodes while management processing has to be performed at the network management station. Any capabilities at network nodes are fixed, embedded by the manufacturer at the network element construction time.

While a protocol-based approach is specific to a management framework, a generic approach based on the client/server model can be achieved through the use of distributed object frameworks as proposed by researchers in the mid-90s. A distributed object framework, with particular influence on the network management area, is the Common Object Request Broker Architecture (CORBA) [2] proposed by the Object Management Group (OMG). Distributed object frameworks, allow the creation of decentralized, static systems. Decentralization is achieved by placing required management logic in network nodes and by creating instances of management objects specific to interested clients. Although distributed object frameworks such as CORBA have succeeded in allowing the development of decentralized systems, they still suffer from a lack of support for programmability, as the management logic located in network nodes is static and cannot be easily altered. The issue of programmability of managed nodes is particularly important for network management systems. The lesson learnt from the deployment of Telecommunications Management Network (TMN) systems was that they were highly complex and suffered from long standardization cycles [3]. The latter means that network managers had to wait several years before a standardization cycle was completed and the required management functionality was embedded in network nodes.

In order to address this problem, recent research has turned into a mobile agents approach, representing an attractive solution for dynamic and programmable network management. Agent mobility can allow the easy programmability of network nodes. This can be achieved by a mobile agent equipped with updated management logic that migrates to the required managed node for execution. Decentralization of a network management task can also be easily achieved through migrating agents that execute near the resources required for their task. In addition to exploiting the mobility of agents, agent autonomy can be used to fine-tune the operation of a network management system as well as for execution of routine tasks on behalf of the user. As an example, a system of mobile agents responsible for a network management task could make autonomous decisions for the initial positioning of mobile agents as well as their possible re-location while executing, allowing their more efficient execution or the balancing of network resources required for the whole task.

A fundamental principle of network management is that any network management task should have a minimum impact on the managed network. Naturally the more advanced capabilities of mobile agent-based systems are also linked with an increase in performance overheads compared to systems based on distributed object frameworks (see [4]). Such performance overheads affecting the managed network are typically associated with mobile agent migration. In an effort to keep these overheads to a minimum many mobile agent-based systems today exhibit single-hop agent migration only or multi-hop migration of small agents only. Another area of increased performance overheads of mobile agents in comparison to static distributed objects is typically associated with their remote communication. This is commonly attributed to the much more dynamic communication mechanisms required to accommodate the remote communication of migrating agents.

This mixed situation of complementing advantages and disadvantages between mobile agent and distributed objects frameworks leads to the direction that the two approaches should ideally coexist. Most important, mobile agents offer easy programmability of managed nodes, while static distributed objects typically offer better performance. A typical network management system involves the collaboration of static software entities with others that would preferably be mobile. As such, real synergy could be achieved in a network management system if stationary entities are provided using static objects that collaborate with mobile agents in a system that combines the best of both. Such a ‘hybrid’ approach to network management based on a combination of Mobile Agents and CORBA entities is presented and assessed herein. While some previous work on the integration of mobile agents and CORBA considered mostly architectural issues [5] [6], in this paper we concentrate on issues of system design and a performance evaluation of the hybrid approach.

In the rest of this paper we begin by presenting (in Section 2) the approaches to agent mobility that influence our hybrid systems' design and capabilities. In section 3 we propose a generic approach to an autonomous and programmable network management system based on a hybrid combination of static CORBA objects and mobile agents. Subsequently, section 4 demonstrates the usage of the hybrid approach through a system for network performance monitoring. Section 5 presents a set of experimental results that show the relative performance levels and confirm the areas of reduced performance overheads of a hybrid approach compared to a mobile agents only system. Finally in section 6 we discuss the lessons learnt and we give pointers to our future research directions.

2 Software Mobility in Network Management

A first clear effort of exploiting software mobility in order to decentralize network management operations was made in 1991 with the work on Management by Delegation (MbD) [7]. The approach is based on mobile code that is sent to a remote node to execute under local or remote control. This was recently revived by IETF as the basis of the Script-MIB proposal [8]. Script-MIB uses the MbD paradigm to define an SNMP-compliant Management Information Base for the delegation of management functions in the SNMP framework. The ITU-T has also standardized a proposal for MbD, called the CMIP Command Sequencer [9]. MbD can be considered a precursor of the Remote Evaluation (REV) paradigm for code mobility described in [10]. In this paradigm, code containing the required logic is pushed along with initial parameters to a remote node, where we have object creation and its execution. This paradigm evolved further into the 'Constrained' mobility model involving software mobile agents [4]. The model was termed 'Constrained' mobility since the agent, upon its creation at a client site, is only allowed to migrate to a single remote server where its execution will be confined. An important aspect of constrained mobility is that a mobile entity is not restrained to be a remote service, as in the case of REV, but instead acts as an autonomous software agent (e.g. choosing its migration node, intelligently collaborating with other agents to achieve its task, etc.). Earlier studies on constrained mobility of agents (e.g. assessments by Bohoris, et al in [4], Simões, et al in [11]) have shown that the model fits well with typical network management requirements in systems that involve long-term management tasks for which programmability of the distributed management logic is required. In addition, since mobile agent migration takes place only once, the performance overheads associated with the model are typically acceptable. Another interesting model of agent mobility is often termed 'Weak' mobility [4]. This involves a mobile agent migrating from the network management station to a number of managed nodes, executing its task in every hop. In weak mobility the migrating agent carries no state information and thus its behavior is not influenced by knowledge acquired in previous hops. The model is particularly suited for management tasks requiring a relatively short period in time. We should also note here that in order to exploit weak mobility in an efficient manner and due to the multiple agent migrations involved, it is important for the size of the migrating agent to be small. The appropriate exploitation of both constrained and weak mobility by suitable mobile agents is illustrated in the approach described in the following section.

3 Proposed Approach

Let’s now consider a network of managed nodes, each hosting a number of CORBA servers containing an executing piece of management logic (Fig. 1). While CORBA servers written in compiled languages are tied to a specific hardware architecture and operating system, a number of CORBA servers are written in an interpreted language such as Java, allowing them to execute in any managed node running a suitable Virtual Machine (VM) and the relevant ORB.

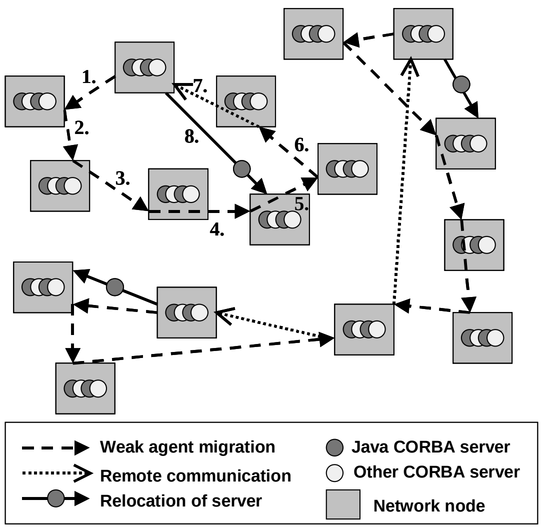

Fig. 1. A system of dynamically positioned Java-based CORBA servers

For the efficient operation of the system, we would like to have these Java-based CORBA servers placed dynamically in managed nodes. This can be achieved with a mobile agent carrying a CORBA server to the appropriate managed node for execution (Fig. 2, step 1). In [12], Liotta et al, presented an algorithm that allows an agent to decide based on an analysis of routing tables on the most efficient node for its migration and execution. The decision mechanisms used in [12] are a result of a theoretical study, assessed through a set of simulation results confirming the usefulness of the process. In the work presented here we enhance those mechanisms by considering a combination of efficiency parameters and present their realization in a system based on current Mobile Agent and CORBA technologies. Since the state of network nodes can change with time, we would like to be informed if there is a more efficient node for execution. For this reason, our mobile agent along with the CORBA server can carry the code of a second smaller mobile agent (Fig. 2, step 1). This small agent can periodically visit (in a weak mobility manner - Fig. 2, step A) a number of alternative nodes and collect local performance information that will allow it to possibly instruct (Fig. 2, step B) a relocation of the CORBA server in a more efficient node of execution (Fig. 2, step 3).

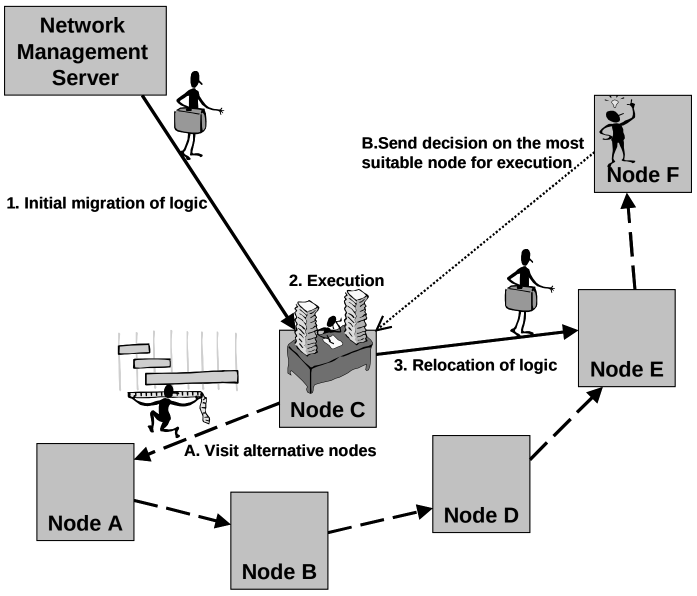
Network Management Server
1. Initial migration of logic
B.Send decision on the most suitable node for execution
Node F
2. Execution
3. Relocation of logic
Node E
Node C
A. Visit alternative nodes
Node A
Node B
Node D

Fig. 2. A typical scenario of operation and re-location of a single Java-based CORBA server

The above discussion allows us to identify that two mobile software entities are required by our approach as well a static object containing the required management logic. In addition, other collaborating static entities can be available in order to formulate a complete management system. In the following section we will demonstrate how the proposed approach can be realized in the context of a system for network performance monitoring.

4 A Prototype Hybrid System for Performance Monitoring

Performance monitoring involves the systematic monitoring of network conditions of a number of crucial nodes in a network. This can be very important for the efficient configuration of network resources and the quick response to critical performance conditions. Our performance monitoring system was specifically developed to support the monitoring of various performance parameters important in an Intranet environment. Based on the approach presented in the previous section we have developed a prototype system for network performance monitoring using a combined CORBA and mobile agents environment. CORBA distributed objects support was provided by Sun’s Java mapping of CORBA included in JDK 1.3.1 [13]. Mobile agents support was provided by IKV's Grasshopper agent platform [14] version 2.2.1. Our performance monitoring system consists of two mobile agents and three CORBA objects as described below (see Fig. 3):

- CORBA Monitor: A CORBA object responsible for performing performance monitoring. It contains all the required management logic for this task involving a combination of metric monitoring and summarization functionality [15].
- CORBA Master: A CORBA object responsible for initiating and controlling the performance-monitoring task. The Master directly sends to the Monitor any requests for reconfiguration of the performance monitoring process. In addition it receives from the Monitor any performance reports or notifications generated.
- Position Agent: A mobile agent responsible for visiting a number of alternative nodes in the network collecting information on their conditions. Based on this, it can make a decision and instruct for the relocation of the performance monitoring task on a more efficient node of execution.
- Worker Agent: A mobile agent migrating along with the code of the Position Agent and the Monitor into a targeted node. Upon arrival it initiates the Monitor object and passes a reference of it to the Master. It is also responsible for periodically creating a Position agent as well as relocating the monitoring task to alternative nodes of execution.
- CORBA Target: A CORBA object available at each network node allowing the Monitor to access raw performance information. The Target allows the Monitor to take passive performance measurements (e.g. Used bandwidth, loss, etc) by wrapping the underlying SNMP support as well as active measurements by providing an 'echo' facility (e.g. for delay, jitter, etc).

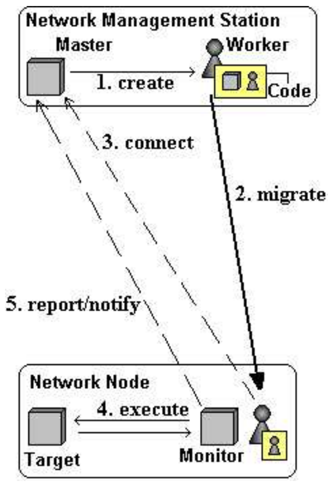

Fig. 3. Initiation of the performance monitoring task

A typical scenario of system operation commences with a request for performance monitoring from a user of the system creating a CORBA Master. The Master is responsible to create a new Worker agent and pass to it any parameters required for the performance monitoring task (Fig. 3, step 1). The Worker, containing the code of a CORBA Monitor and a Position agent, subsequently migrates to the chosen network node. Upon arrival it initiates a CORBA Monitor and passes its reference to the Master so that the two CORBA objects can collaborate directly. The Monitor initiates its task with periodic requests for 'raw' performance information (counter type values) provided by the CORBA Target. Based on this information the Monitor performs metric monitoring for a number of performance parameters (e.g. Used Bandwidth, Loss, etc), checks the thresholds set and gathers the information produced in order to generate reports. During this task, the Monitor remotely sends to the Master such reports on performance conditions in a scheduled manner (e.g. every 1 hour) as well as notifications in real time when a performance threshold is triggered.

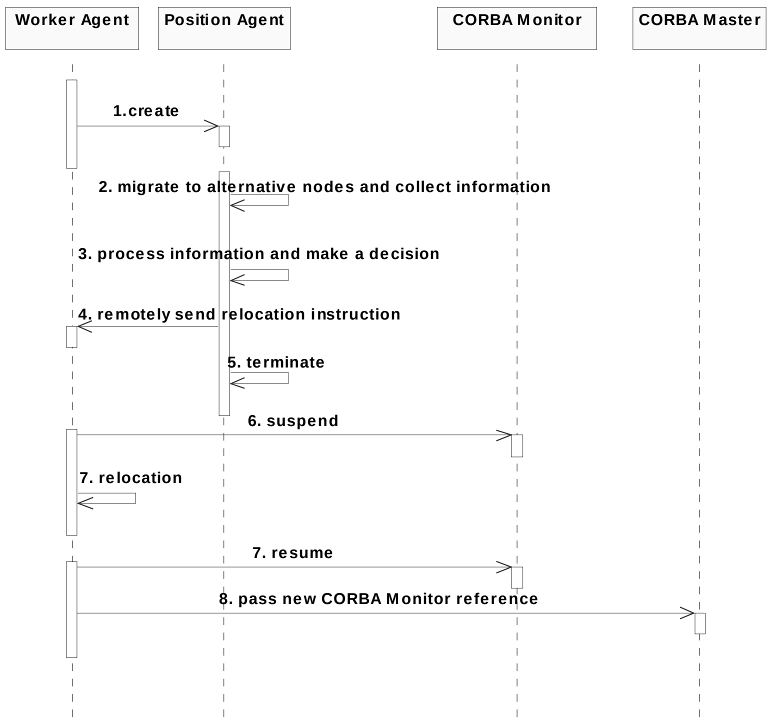
Worker Agent
Position Agent
CORBA Monitor
CORBA Master
1.create
2. migrate to alternative nodes and collect information
3. process information and make a decision
4. remotely send relocation instruction
5. terminate
6. suspend
7. relocation
7. resume
8. pass new CORBA Monitor reference

Fig. 4. A typical relocation scenario

Further to this initial scenario, the Worker agent periodically creates a Position agent to examine the possibility of relocating in a more efficient node (see Fig. 4). The created Position agent visits in a weak mobility manner a number of alternative nodes and collects information on their conditions (e.g. CPU load, memory usage, delay, number of executing threads, etc) at each hop. When the Position agent finishes its task at the last visited node it analyses the collected information and makes a decision on whether relocation is beneficial. If so, it remotely instructs the Worker on the suitable relocation node to migrate and finally terminates. The Worker subsequently suspends the currently running monitor and migrates to the designated node. Upon arrival it initiates the CORBA monitor that can resume its operation from the previously suspended session information. The Worker finally needs to notify the Master of the new Monitor reference in order to allow again the direct communication between the two CORBA objects.

The system's approach has allowed us to exploit a number of mobile agent capabilities allowing for dynamic and programmable performance monitoring. In the section that follows we will present experimental results showing the improvements in performance introduced by the collaboration of agents with static distributed objects.

5 Evaluation and Assessment

5.1 Environment and Method

In our evaluation we are interested to highlight the performance improvements in the managed network associated with a hybrid system as compared to a Mobile Agents only system. Additionally contrasting the hybrid system with a CORBA only system will reveal the levels of any additional overheads of our approach. Within this scope we have developed two comparison systems based on Grasshopper Mobile Agents only and CORBA only. The important areas of performance overheads in the network involve the tasks of agent migration as well as remote communication of the various software entities. In this context and for the three available systems, measurements were taken for the following typical system operations:

- The scheduled remote transmission of a performance report (see Fig. 3, step 5) from the monitoring entity at the managed node to the Master at the network management station. A performance report contains a list of all the performance monitoring information produced by the CORBA Monitor. For our measurements we considered reports containing a list of 25, 50, 75 and 100 'double' numbers, reflecting on information gathered during the metric monitoring process.
- The Worker agent migration carrying the management logic to a managed node as well as the Position agent migration to an alternative node (see Fig. 3, step 2 and Fig. 2, step A).

For these operations we have taken the following measurements:

- Traffic measurements: Taken using the tcpdump utility with the sizes reported reflecting on the total payload at TCP level.
- Response times measurements: Taken using the System.currentTimeMillis() method included in the API of Sun's JDK. The measured values of response times reported represent the "steady-state" running costs, excluding the initial setup costs.

In addition to performance overheads in the network we consider the resources required in the managed node by each system. In this direction we have measured the memory requirements of each system at a managed node. The measurements were taken using the totalMemory, and freeMemory methods of the java.lang.Runtime class. The first method provides the total amount of memory allocated by the Java Virtual Machine (JVM). The second one returns an approximate value of the amount of memory left free inside the JVM i.e., memory available for future object allocation. The difference of these two values provides the amount of memory required by the performance monitoring system under evaluation including any platform support objects.

All measurements were taken using a testbed of Linux workstations with homogeneous features (Redhat 7.1, Intel Celeron 466MHz and 64MB of RAM), connected to a 100Mbps Ethernet network.

5.2 Results

The response times of the three systems for the operation of transmitting a remote report of 25, 50, 75 and 100 ‘double’ numbers can be seen in Fig. 5. Note once more that in the hybrid system remote communication between the Monitor and the Master objects takes place through the CORBA facilities. Thus, for this operation the hybrid system has the same performance overheads with a plain CORBA system. Fig. 5 shows that the hybrid system performs about 5 times better than the mobile agents only system.

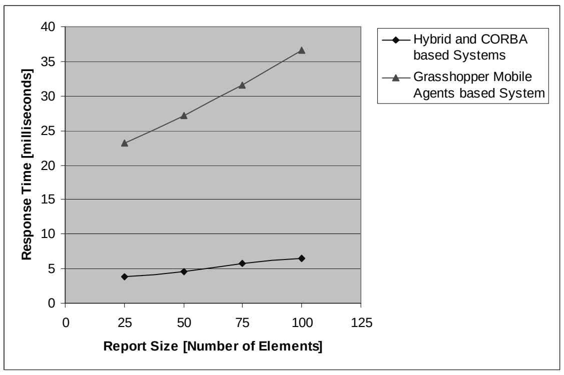

Fig. 5. Response times for the transmission of performance reports

The traffic that the three systems incurred in the network for the operation of transmitting a remote report of 25, 50, 75 and 100 ‘double’ numbers can be seen in Fig. 6. As before, for this operation the hybrid system has same performance overheads with a plain CORBA system. Fig. 6 shows that the hybrid and CORBA only systems incur constantly less traffic than Grasshopper Mobile Agents with the gap increasing as the number of reported elements increases.

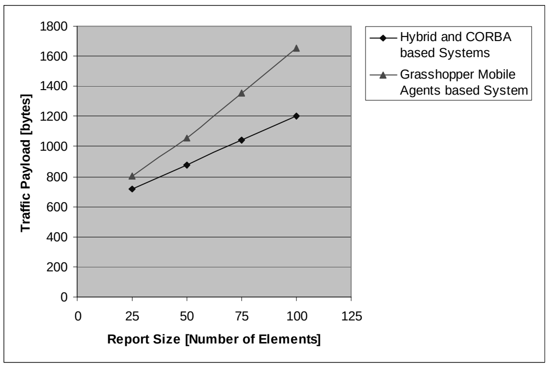

Fig. 6. Traffic incurred for the transmission of performance reports

Software migration is an additional overhead for the two systems involving mobile agents. In the table below the migration times and traffic incurred by the mobile agents of the hybrid and mobile agents only systems are reported.

	Hybrid System - Worker Agent	Mobile Agents System - Worker Agent	Position Agent
Response Time [msec]	1705	1524	658
Traffic Payload [bytes]	3282	3096	1316

The Worker Agent of the hybrid system gave slightly higher overheads compared to the Worker Agent of the Mobile Agents only system. This is due to the additional CORBA specific code required in the entities of the hybrid system. The Position Agent is common for both systems and much smaller than the Worker Agent resulting in significantly smaller performance overheads that make it suitable for the Weak mobility model. For comparison with the performance overheads of the mobile agents reported above, in a CORBA distributed objects system the creation of a distributed object through a factory requires typically less than 15ms to complete and incurs around 500 bytes of traffic.

Finally, the results regarding the JVM memory usage at a managed node can be seen in Fig. 7. The hybrid approach requires the most amount of memory as it combines both CORBA and mobile agent entities and supporting platform facilities. We have to note here that our reported values of total memory regarding the hybrid system involve the total of two JVMs used, one for the CORBA target object and one for the Grasshopper execution environment including the performance monitoring entities within.

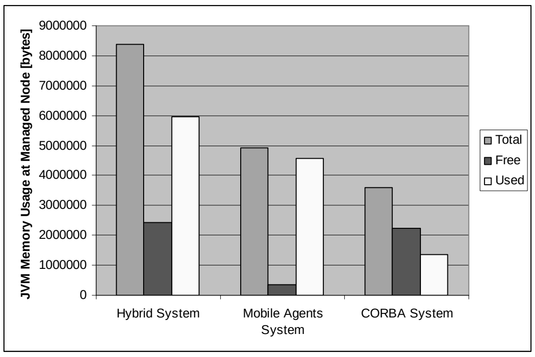

Fig. 7. JVM memory usage at a managed node

6 Conclusions

In this paper we presented the approach, design and evaluation of a hybrid network management solution based on a combination of mobile agents and CORBA. A demonstrating system for network performance monitoring illustrates the manner that the approach exploits mobility and autonomy of agents as well the good performance levels offered by static CORBA objects. An important result arising from this work is that we can easily enhance an existing CORBA system with mobile agent capabilities, through a small alteration of the static CORBA objects at the network management server and no changes necessary to the CORBA servers code executing in the managed nodes. In our case study for performance monitoring small alterations were made to the CORBA Master in order to create and communicate with the Worker Agent while the code of the CORBA Monitor required no modifications. Constrained mobility of agents provides an easy and effective approach to programmability of management logic at network nodes. Through our practical experience we have identified as a rule of thumb that agents exhibiting constrained mobility can be used to replace CORBA factories located at a managed node. Programmability ensures a flexible system that evolves quickly along with changing network management requirements, an element that was missing from the TMN systems of the mid 90s and hindered their success. As shown by our experimental results the hybrid approach resulted in a significant reduction of performance overheads in the network for typical management operations compared to a mobile agents only system and in similar levels with CORBA. Naturally, agent migration is an important additional overhead compared to CORBA systems but the constrained mobility aim of long-term execution in a network node helps to ease the migration overheads concerns. In addition in our proposed approach weak mobility of agents was exploited efficiently by ensuring that the size of the Position Agent was small. Our hybrid system approach also exploits the autonomous nature of mobile agents allowing us to fine-tune the system’s operation by efficiently positioning our agents and balancing the usage of resources in the managed network. While our current work was a first step in exploiting agent autonomy our future work will focus in the direction of more sophisticated autonomous decisions and the detailed evaluation of their significance as compared to a plain delegation approach.

References

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

[2] Object Management Group, "The Common Object Request Broker: Architecture and Specification (CORBA)", Version 2.0, 1995.

[3] G. Pavlou, G. Mykoniatis, J. Sanchez, "Distributed Intelligent Monitoring and Reporting Facilities", IEE Distributed Systems Engineering Journal (DSEJ), Special Issue on Management, Vol. 3, No. 2, pp. 124-135, IOP Publishing, 1996.

[4] C. Bohoris, A. Liotta, G. Pavlou, "Software Agent Constrained Mobility for Network Performance Monitoring", In the Proceedings of the 6th IFIP Conference on Intelligence in Networks (SmartNet 2000), Vienna, Austria, ed. H.R. van As, pp. 367-387, Kluwer, September 2000.

[5] F. G. Chatzipapadopoulos, M. K. Perdikeas, I. S. Venieris, "Mobile Agent and CORBA Technologies in the Broadband Intelligent Network", IEEE Communications Magazine, Vol. 38, No. 6, pp. 116-124, June 2000.

[6] M. H. Guiagoussou, R. Boutaba, M. Kadoch, "A Java API for Advanced Faults Management", IEEE/IFIP International Symposium on Integrated Network Management (IM'01), pp. 483-498, 2001.

[7] G. Goldszmidt, Y. Yemini, "The Design of a Management Delegation Engine", Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Santa Barbara, CA, October 1991.

[8] D. Levi, J. Schonwalder, RFC2592 – "Definitions of Managed Objects for the Delegation of Management Scripts", The Internet Society, May 1999.

[9] ITU-T Rec. X.753, "Command Sequencer", 1997.

[10] A. Fuggetta, G. P. Picco, G. Vigna, "Understanding Code Mobility", IEEE Transactions on Software Engineering, vol. 24, no. 5, pp. 342-361, 1998.

[11] Paulo Simões, João Rodrigues, Luís Silva, Fernando Boavida, "Distributed Retrieval of Management Information: Is it About Mobility, Locality or Distribution?", In the Proceedings of the Network Operations and Management Symposium 2002, Florence, Italy, 15-19 April, 2002.

[12] A. Liotta, G. Pavlou, G. Knight, "Exploiting Agent Mobility for Large Scale Network Monitoring", IEEE Network, special issue on Applicability of Mobile Agents to Telecommunications, Vol. 16, No. 3, IEEE, May/June 2002.

[13] Sun's Java Development Kit, http://java.sun.com/j2se/

[14] IKV++, Grasshopper Mobile Agent platform, http://www.grasshopper.de.

[15] ITU-T Rec. X.739 – "Metric Objects and Attributes" (1992) and ITU-T Rec. X.738 – "Summarization Function" (1993).