Software Agent Constrained Mobility for Network Performance Monitoring

C. Bohoris, A. Liotta, G. Pavlou
Center for Communication Systems Research
School of Electronic Engineering and Information Technology
University of Surrey, Guildford, Surrey GU2 5XH, UK
{C.Bohoris, A.Liotta, G.Pavlou}@eim.surrey.ac.uk

ABSTRACT

During the recent years of research on mobile agents, significant effort has been directed towards the identification of models of agent mobility suitable for network management applications. Also, a lot of research work is currently being carried out trying to provide an assessment of mobile agent frameworks used to build agent-based network management systems. In this paper we clarify three different models of agent mobility, present a mobile agent-based performance monitoring system that exhibits the "constrained mobility" model, and discuss its practical use for dynamically programming network elements. The implementation of this system is presented and compared with static object approaches. Furthermore we provide a performance evaluation of the mobile agent based system as it compares with Java-RMI and CORBA distributed frameworks, in order to assess the advantages, along with the overheads, of agent solutions.

KEYWORDS

Mobile Agents, Constrained Mobility, Performance Monitoring.

1. INTRODUCTION AND BACKGROUND

Network management has been the subject of intense research over the last decade, with the relevant progress being twofold: on the one hand, architectures and algorithms for solving management problems have been devised; and on the other hand, different management technologies have been proposed and standardized. From the protocol-based approaches of the early 90's, exemplified by the Simple Network Management Protocol (SNMP) [1] and OSI Systems Management (OSI-SM) [2], the focus moved to distributed object-based approaches in the mid to late 90's, exemplified first by the Common Object Request Broker Architecture (CORBA) [3] and later by Java's Remote Method Invocation (Java-RMI). More recently, the

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35522-1_37
H. R. van As (ed.), *Telecommunication Network Intelligence*

focus seems to be shifting back to protocol-based approaches the emerging Directory Enabled Networks (DEN) framework.

The paradigm of moving management logic close to the data it requires is a technique that has been conceived early in the evolution of management architectures, the relevant framework known as "management by delegation" [4]. Subsequent research showed the applicability of this concept in the context of OSI-SM [5] while a similar approach was subsequently standardized, the Command Sequencer Systems Management Function (SMF). More recently, the same concept has been proposed in the context of SNMP through the Scripting MIB. While such approaches are specific to the respective management frameworks, delegation in the context of general distributed object frameworks is achieved through *object mobility*. Mobile objects are usually termed *mobile agents* and when they act through emerging behavior in the Artificial Intelligence (AI) sense, they become *intelligent agents*. Mobility and intelligence are though orthogonal properties.

The emergence of mobile agent frameworks has led many researchers to examine their applicability to network management and control environments. [6] considered first code mobility and presented a taxonomy of the relevant aspects. [7] considered mobile agents in the context of the Intelligent Network (IN) and proposed an agent-based architecture for "active" IN service control. [8] discussed the general issues of using mobile agents for network management while a number of other researchers have attempted to use mobile agents in specific network management case studies. It is believed that mobile agents can provide better solutions at least to performance and fault management problems, given the large amount of data that needs to be moved around in respective solutions based on traditional approaches.

Mobile agents may move around the network from node to node and clone / destroy themselves according to their intelligence. We term this situation "strong mobility" and it is this property that has not yet been exploited to good effect in network management. An alternative possibility for mobile agents is to move from node A to B, typically guided by a "parent" stationary agent, and stay there until their task is accomplished. We term this situation "constrained mobility" and we believe it is this approach that can be readily exploited in management environments. In this case, instead of predicting the required functionality, standardizing and providing it through static objects in network elements, mobile agents could support it in a dynamic, customizable fashion. The key advantage in this case is that the target node needs only to provide the required "bare-bones" capability which could be dynamically augmented through mobile agents, with the mobile agent logic able to change to reflect evolving requirements over time. Such a possibility would obviate the use of functionality such as the OSI-SM

Systems Management Functions (SMFs) and similar capabilities provided in SNMP.

In the work described in this paper we are trying to evaluate the use of mobile agents for network performance monitoring, assuming a constrained mobility paradigm in which a mobile agent is sent to execute and monitor information within a network element. The latter can be managed through a collection of static agents that offer similar capabilities to a OSI-SM or SNMP Management Information Base (MIB). The evaluation is twofold: first, we are interested in assessing the usability of a mobile agent platform as opposed to a static object platform such as CORBA or Java-RMI, and in particular the agent customization aspects; and second, we would like to examine the performance implications of using mobile agents in order to assess if the provided flexibility is potentially outweighed by the additional performance overhead. This work has been partly carried out in the context of the MIAMI ACTS project (Mobile Intelligent Agents in the Management of the Information infrastructure) [9], which examines the impact and possibilities of using mobile agent technology for network and service management.

The rest of this paper has the following structure. In Section 2 we summarize briefly the way in which performance monitoring is supported through generic but predefined functionality in the context of OSI-SM, SNMP and CORBA-based management systems. In Section 3 we examine three models of agent mobility that can be applied in network management applications. In Section 4 we concentrate on the performance monitoring system and present our agent implementation based on constrained mobility. In Section 5 we present an evaluation and assessment of the performance monitoring system and in Section 6 we present a summary and conclusions.

2. STATIC PERFORMANCE MONITORING

Performance management is one of the management functional areas identified initially in OSI Systems Management (OSI-SM) [2]. It addresses the availability of management information in order to be able to determine network load under both natural and artificial conditions. It supports the collection of performance information periodically in order to provide statistics and allow for capacity planning activities. Performance management needs access to a large quantity of dynamic network information. An important issue is to provide this information to management applications with a small impact on the managed network. A key requirement is the ability to convert raw traffic information to traffic rates with thresholds applied to them so that Quality of Service (QoS) alarms can be generated. An additional requirement is the periodic summarization of

a variety of performance information for trend identification and capacity planning purposes.

QoS management applications monitor performance aspects both "within" the network and at "edge" nodes where customer services are offered, trying to identify potential performance degradations. They may subsequently trigger the reconfiguration of parts of the network in order to alleviate congestion e.g. by changing the routing strategy, re-allocating resources such as bandwidth to trails, etc. Monitored aspects of services may include the service availability, the supported capacity in terms of available bandwidth and the end-to-end delay. In the case of a "leased line" service with guaranteed QoS, e.g. as part of a Virtual Private Network (VPN) service, its availability may be affected by faults while the available capacity and delay may be affected by network congestion when the provided bandwidth is multiplexed. In general, performance management is coupled with both fault and configuration management.

A simplistic approach for collecting the required performance information is through periodic polling. In this case, the collected raw data is processed either at a centralized Network Management Station (NMS) or at Element Managers (EMs) which may form part of a hierarchical management system e.g. following Telecommunications Management Network (TMN) [10] principles. The problem with this approach is that it generates substantial management traffic and, subsequently, does not scale (it should be mentioned though that the generated traffic is smaller in the hierarchical compared to the centralized case). An alternative approach is to delegate monitoring activities to the network elements, reporting only QoS alarms or summarized reports to higher-level managers. The OSI-SM Metric Monitoring [11] and Summarization [12] systems management functions have addressed this requirement through generic Support Managed Objects (SMOs), which need to be provided in managed network elements. Facilities similar to metric monitoring have been subsequently provided in SNMP environments, initially in the Remote Network Monitoring (RMON) [13] specification. In addition, similar facilities could be provided in CORBA-based network elements.

The problem with such generic functionality is that it needs to be first researched, standardized, implemented and eventually deployed in network elements; this process typically takes a long time. In addition, any modification, e.g. for providing more sophisticated features that were not thought out in advance, needs to go through the full research, standardization and deployment cycle. For example, [14] identified additional functionality that combines the capabilities of metric monitoring and summarization objects in a powerful fashion but such additions need to go through a new standardization cycle. The specification of the OSI-SM systems management functions took a long time and is partly responsible for the perceived

complexity and lateness in the deployment of OSI-SM-based network elements.

Mobile agents could provide similar facilities in a dynamic fashion, allowing for network elements with bare-bones *real resource* only and no *support* management information. Additional generic capabilities could be provided through mobile agents which would be sent to execute within a network element. The behavior of those agents could be altered dynamically at any time. The flexibility provided to management applications would be enormous, since they could now "customize" network elements for performance and other management activities according to their requirements and would not be restricted by the available standardized facilities. On the other hand, network elements should be able to host mobile agents through suitable platform infrastructure and real resource managed objects should be realized as static agents. The performance implications of using mobile agents universally are addressed later in this paper.

3. AGENT MOBILITY IN MANAGEMENT

The problem of identifying the features that distinguish an agent from a common computational entity has raised controversial arguments for nearly a decade and only recently those features have been identified. Theoretical studies on agents and artificial intelligence have come to the conclusion that a computational entity can be regarded as an agent if it exhibits some of the following properties: social ability, autonomy, reactivity, proactivity, adaptability, persistency, and ability to learn, communicate, co-operate, and move [19].

Other research work focusing on the practical application of agent theories tends to characterize agents with a subset of the above properties. For instance, Mobile Agents are commonly defined as computational entities that act on behalf of some other software entity, exhibit some degree of autonomy, and are particularly featured with migration capability.

The chief benefits that agent mobility can bring into the network management arena, for each of the five management functional areas, are highlighted in [8]. Some of those benefits include reduction in network traffic, efficient utilization of computational resources, support for heterogeneous environments, and increased flexibility. Nevertheless, the use of mobile agents does not come without costs. In particular, code migration incurs additional traffic into the network, absorbs considerable resources from the agent hosts, and is associated with migration delays of the order of seconds or even tens of seconds, depending on the agent configuration and functionality [20] (see also Sec.5.2 below). In some cases the agent migration overheads outweight their benefits and make this approach inconvenient. It is therefore important to grasp the various aspects of agent

mobility and to relate them to network management in order to identify those aspects that are particularly beneficial.

In the following subsections we define three different types of agent mobility, ranging from the simplest, light-weight form of mobility to the most heavy-weight one. For each case we elaborate on its benefits and limitations in relation to network management, identifying advantageous scenarios.

3.1 Constrained Mobility

One of the most elementary forms of code mobility is defined in [6] as Remote Evaluation (REV), after the pioneering work described in [21]. In REV, an application in the client role can dynamically enhance the server capability by sending code to the server. Subsequently, clients can remotely initiate the execution of this code that is allowed to access the resources collocated within the server. Therefore, this approach can be seen as an extension of the client-server paradigm whereby a client in addition to the name of the service requested and the input parameters can also send code implementing new services. Hence the client owns the code needed to perform a service, while the server offers both the computational resources required to execute the service and the access to its local resources.

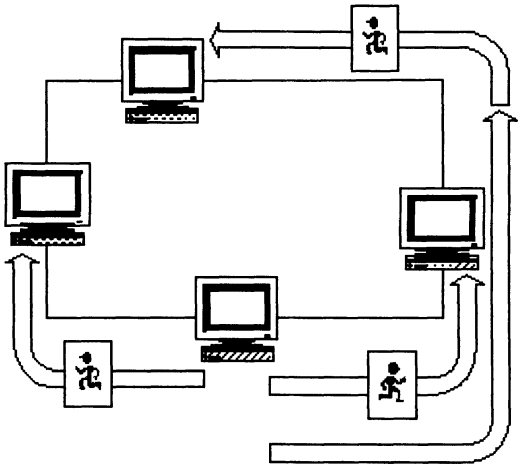

Figure 1 – Constrained mobility. The agent is created and initialised by a client application and is then shipped to an agent host. The agent execution is then confined to this host.

A natural evolution of the REV model involves sending code possessing one or more of the above mentioned agent features – e.g. mobility and autonomy. This type of agent mobility can be defined as *constrained*

mobility since the agent, upon its creation in a client site, is only allowed to migrate to a remote server where its execution will be confined.

When constrained mobility is adopted in management the agent is created by a client acting in the manager role and is, then, dispatched to a target network element acting in the server role (Figure 1).

This approach is particularly suited to dynamically programming or upgrading network devices. In this case, agents do not need to be particularly sophisticated. In a simple scenario these agents do not even need to have any sort of autonomy – e.g. they do not need to have the ability to select their target network element, since the manager can provide this information – and could be as simple as collections of objects that can be executed in a remote virtual machine. Therefore, mobility degenerates into a simple dynamic mechanism to efficiently deploy or upgrade network protocols or services.

Agent deployment overheads, namely *deployment traffic* and *delay*, represent the drawbacks of general MA approaches. In constrained mobility the agent does not need to incorporate complex migration features and, as a result, its size and the incurred network traffic are minimal if compared to the other forms of agent mobility (see sections below). Similarly, it will not be necessary to use general purpose MA platforms – usually associated with heavy-weight migration mechanisms [20] – and, thus, the agent migration time can be considerably reduced.

In conclusion, constrained mobility is a particularly well-suited mechanism to dynamically program network elements. It can outperform traditional centralised management for data-intensive tasks and when high degrees of semantic data compression need to be achieved – e.g., through data aggregation or analysis. Constrained mobility is typically advantageous to perform off-line analysis of bulk data and, more generally, to implement tasks whose duration is at least comparable with the overall agent deployment time.

3.2 Weak Mobility

Similarly to constrained mobility, in weak mobility the agent is created and initialised by a client application and is, then, shipped to an agent host. However, 'weak' MAs are not confined to this host since they are meant to perform the same task in more than one location (Figure 2). Weak MAs do not retain any knowledge of the data processed or of the actions performed in previously visited hosts and, consequently, they can only implement tasks in which this information is not required.

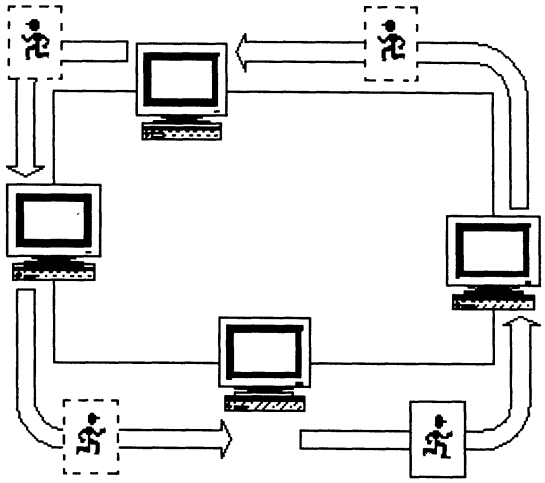

Figure 2 – Weak mobility. The agent task involves the visitation of many hosts, but no information gathered in previous visits is preserved.

A convenient use of weak mobility is for dynamic decentralisation of management tasks that are otherwise performed in a centralised fashion. The agent is delegated part of the management responsibility and will incorporate functionality such as procedures aimed at data semantic compression or aggregation.

A trivial example showing the main advantages of weak mobility is the case in which the management station has to search for a single value in a table, a data structure typically used to store information inside devices. In SNMP management the whole table has to be transferred from the remote element to the management station, where the table rows are searched for the value. Hence, large tables will incur heavy unnecessary traffic into the network and will result in computational overload on the management station.

A more efficient approach is adopted by OSI management which supports remote scope and filtering operations. Thus, the searching routine is executed in the device and, consequently, only the retrieved value is transmitted to the manager. The drawback of this approach is that routines such as the one implementing scope and filtering have to be hard coded into the network elements which tend do become complex since a large number of routines need to be implemented. Even worse the introduction of new routines requires a cumbersome standardisation process and their deployment needs complex software upgrade.

If constrained mobility was to be used the agent incorporating the search routine would be shipped to the network device, where it would retrieve the requested value from the local table and issues this value to the manager.

This solution addresses the shortcomings of the OSI approach, retaining its performance benefits. However, constrained mobility is not suitable in the more general case in which tasks analogous to the searching routine are to be run on multiple network elements. In fact, as the number of network elements to be searched grows, the management station will be overloaded and the network capacity around it will be saturated by the simultaneous generation and transmission of the agents.

Tasks characterised by a relatively short duration and involving multiple network elements are more efficiently implemented according to the weak mobility model. Since only a single MA is issued by the manager, both network and computing resources around the manager station are preserved. Weak mobility is, thus, suitable to collect on-line data and to perform simple control and configuration tasks from several network elements.

Therefore, weak mobility can lead not only to a reduction in network load but also to a more even utilisation of processing resources through a dynamic serialised distribution of simple management tasks.

3.3 Strong Mobility

With strong mobility agents in addition to being able to access and process data from network elements can also accumulate information and preserve it upon migration (Figure 3). This feature allows for the implementation of more elaborate tasks in which the agent operations depend on data gathered in previously visited hosts. In other words, the agent operation can be altered by the data.

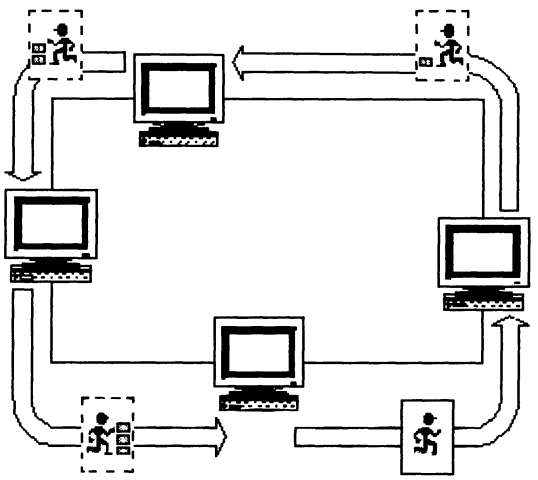

Figure 3 – Strong mobility. The agent task involves the visitation of many hosts and the preservation of information gathered in previous visits.

In management, strong mobility is particularly well suited to configuration tasks and to data-intensive tasks involving data aggregation from highly distributed network elements and on-line data analysis. A simple example is a task involving the collection of utilisation information from a relatively large number of network elements. In a traditional SNMP-based system the management station has to poll every single element in order to collect the required raw performance information before it can produce a useful utilisation rate. This may incur heavy traffic and may even not be acceptable if the number of the elements is too large. OSI management offers a more efficient mechanism for obtaining the utilisation rates, as this is done locally at the network element, but still requires further aggregation of this information at the management station.

Constrained mobility does not suit this task since it would require the deployment of a number of MAs equal to the number of network elements. Each MA would typically be executing for a time negligible with respect to its deployment time and, then, the agent deployment overheads would be unacceptable.

On the other hand, weak mobility would represent an inefficient imitation of the OSI approach. In fact, the agent would have to visit sequentially all the elements and report the local utilisation rates back to the manager before migrating to the next element. Then, we still would not achieve a total decoupling between network element logic and network management logic, since the manager would be still concerned with the collection of data from each element and with its aggregation.

Such decoupling can be achieved through strong mobility, in which case the agent will preserve the utilisation rates of all previously visited elements and will then be able to perform a further level of data aggregation independently from the manager.

The main drawback associated with strong mobility is the agents' size. 'Strong' agents need to incorporate more intelligence that others and tend to be larger in size. More critically, the agents' size can vary significantly depending on the amount of information that has to be preserved during migration. It is, therefore, important to design the agents in such a way to limit their size variations – e.g., by allowing only semantically compressed information to be stored.

In conclusion, strong mobility can be employed to implement data-intensive tasks requiring aggregation of information from different network elements.

4. CONSTRAINED MOBILITY IN A PERFORMANCE MONITORING SYSTEM

In this section we elaborate on the model of constrained mobility by describing its application in a mobile agent based performance monitoring system. As mentioned earlier the constrained mobility model is particularly suited to network management tasks that require a relatively long period of time to execute, when dynamic programming of network devices is required, or when a large number of data collected is intended for off-line analysis. An effective performance monitoring system typically possesses all of the above characteristics making the constrained mobility agent-based approach particularly suitable to it. A user of such a system typically wishes to gather performance information from a number of different machines in the network, receive performance reports on a scheduled basis and on-the-fly notifications when a performance threshold is triggered. The user can analyse the performance reports and notifications obtained to determine utilization trends, isolate performance problems, and possibly solve them before they adversely impact network performance. In this way performance monitoring can also aid in capacity planning and in the provisioning of a consistent level of service to all users of a network.

4.1 System Realisation Based on Constrained Mobility

A typical scenario of operation for an agent-based system is actually very similar to a scenario followed by a system based on static distributed objects. Initially, in the client side of both systems, a static "master" entity is responsible for accepting a user request and initiating the management service. This entity corresponds to the Network and Element Management Layer (NML and EML) functionality in terms of the TMN model.

In the distributed objects system an object with that role would send a remote request for the initiation of a management service to an object located in a remote server machine. These objects located in the server side of the system are responsible for fulfilling the service request using the required management logic, which pre-exists there in a fixed manner. The server objects can also remotely communicate with the "master" object in the client side in order to report important information.

In the case of the agent-based system (Figure 4) using the model of constrained mobility, a "master" agent in the client side will initially create a mobile agent that owns the required management logic to fulfil the service request. This mobile agent will then migrate to the remote server machine where it can have local access to the resources required to perform its task. A static "target" agent that pre-exists in the server side is responsible for providing access to those resources to such a mobile agent on request. The functionality provided by this "target" agent at this level corresponds to the TMN Network Element (NE) level.

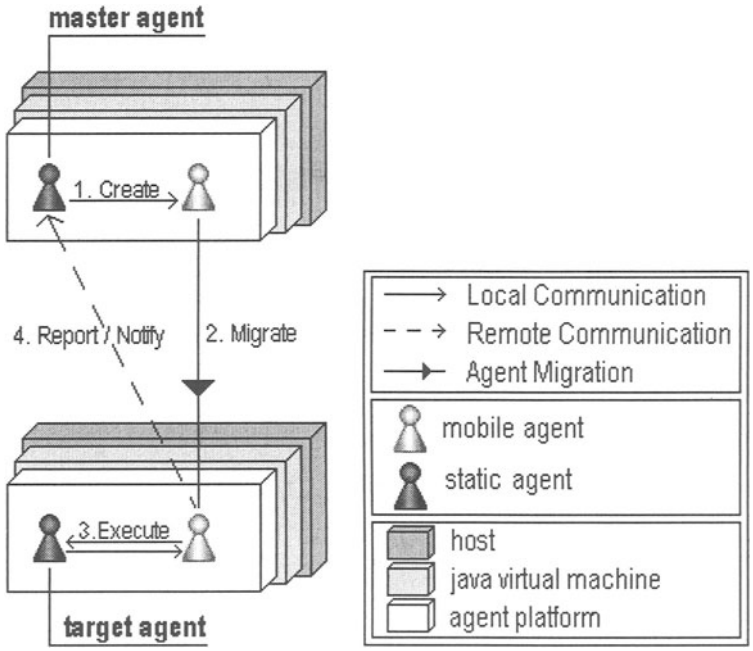

Figure 4 – A mobile agent based network management system using the model of constraint mobility.

The performance monitoring system we developed is based exactly on the above design scenario of an agent based network management system using constrained mobility. A user request for performance monitoring of a remote machine will initially be passed to the "master" agent, which will create a mobile performance monitor agent. This mobile agent will be provided with the specific monitoring parameters as set by the user and will then migrate to the remote machine. Upon reaching its destination the MA will contact a "target" static agent that pre-exists there and is responsible for providing "raw" performance information on request. The performance monitor agent will process this information to obtain rates of utilisation and loss. The performance monitor agent will remotely send reports of the results gathered

back to the "master" agent in the client side, on a scheduled basis. It can also send notifications in real time when an applied performance threshold is triggered. The performance monitor agent provides the functionality of a metric monitor and a summarisation object as specified in the X.739/X.738 standards.

4.2 System Implementation

For the development of our mobile agent-based performance monitoring system we used purely the Java programming language, with all system classes built using Sun's JDK version 1.1.7b. The Grasshopper agent platform version 1.2 was also used, providing a simple execution environment for agents, and an API covering all the required basic agent functionality.

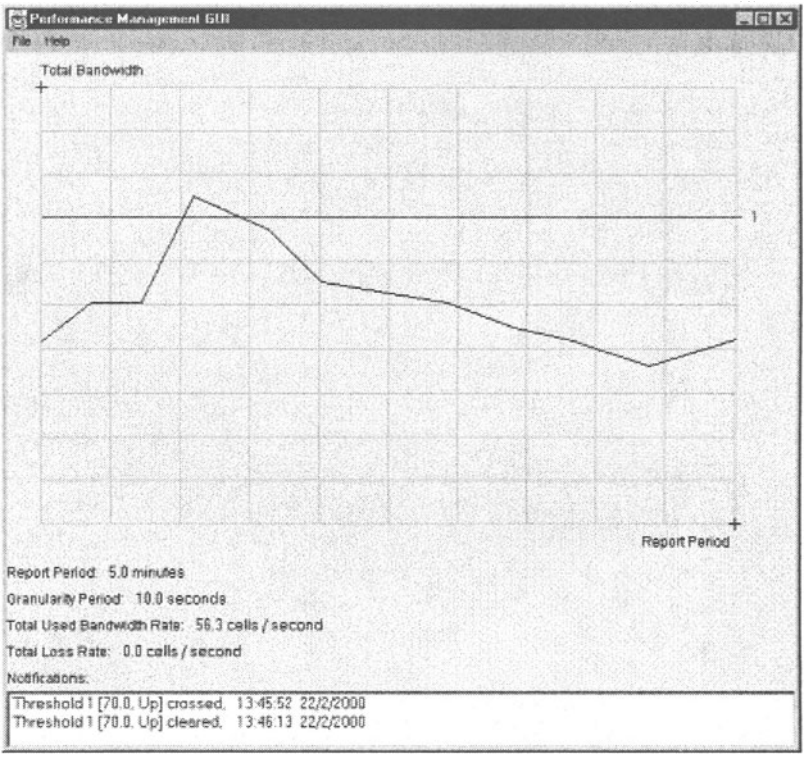

Figure 5 – The performance monitoring system graphical user interface. The graph of utilisation rates gathered goes over the threshold line for a while, indicating that this threshold was exceeded as it can also be seen from the notifications displayed in the status window, located in the lower part of the picture.

The whole development work was done under Sun's Solaris version 7 of the UNIX operating system. For the development of the "target" agent, the AdventNet SNMP version 2.0 libraries were used in order to obtain raw performance information by querying an SNMP agent. The system was built to operate in two different modes for the monitoring of IP and ATM traffic, respectively. All information gathered during the performance monitoring process appears on a graphical user interface in the client's machine, as shown in figure 5.

5. EVALUATION AND ASSESSMENT

While in the previous section we showed how mobile agent technology can be used instead of a static distributed object technology for building hierarchical management systems with the additional advantage of dynamic customisation and object migration, in this section we look at the performance implications of using mobile agent technology. As such, we decided to design and build two additional versions of the system, using Java-RMI and CORBA respectively as static distributed object platforms. The reason we chose Java-RMI is that the Grasshopper platform also uses Java-RMI's JRMP protocol (as well as a proprietary protocol), so we are able to see the precise overheads incurred by the mobile agent support infrastructure. In addition, the comparison with CORBA allows us to draw conclusions on the overheads of mobile agents platforms compared to an emerging distributed object technology for network/service management.

In the case of the Java-RMI/CORBA based performance monitoring system, a "master" object located in one machine sends a request for performance monitoring to a "factory" object located at a network element. When a request arrives to the "factory" object it is responsible to locally create a new instance of a performance monitor object that will perform performance monitoring and summarization functions. A "target" object is also located at the network element and provides raw performance information. The functionality and algorithms in all systems were identical so that we could directly compare the different approaches. It should be noted though that in the case of a static distributed object approach the functionality of the performance monitor object is static and cannot be altered, in a similar way to OSI-SM and SNMP support object facilities. Finally, we chose to use CORBA with the Java mapping for reasons of uniformity and we used the Sun Microsystems openly available version of CORBA with the Java mapping.

5.1 Environment and Methodology

Three performance monitoring systems were used, all containing the same functionality, written using Grasshopper mobile agents, Java-RMI and CORBA. For the evaluation we considered four different cases, one for every system running over its standard communication protocol, and an additional case for the mobile agent Grasshopper system running over the JRMP protocol. We were interested in measuring the following aspects for each case. First, we measured remote invocation response times. Timestamps were taken using the *currentTimeMillis* method of the *java.lang.System* class. A list of 25 elements was remotely transferred 100 times between two objects located in different machines, each time measuring the total time and finally calculating the average and standard deviation of these measurements. The same procedure was repeated while increasing the number of elements in the list to 50, 75, and 100. This operation in fact models the periodic summarization reports generated and remotely sent by the performance monitor mobile agent.

For the same experimental cases, we measured the TCP packet sizes using the tcpdump program that originated at the Lawrence Berkeley laboratory, reporting the total payloads at the TCP level. All these measurements where taken using two different machines over a lightly loaded 100 Mbit/sec Ethernet in the role of the management network with the following specification: Sun Microsystems Ultra-10, 256MB of memory, Sun's Solaris 2.5.1 version of UNIX. Finally, we measured the additional overheads incurred during MA deployment, namely the agent deployment time and the total amount of bytes incurred during the agent transmission from the manager to the network element.

5.2 Response Times

The response times of management operations for each of the three performance monitoring systems have been considered. We examined the performance aspects of remotely invoking operations between two objects located in different machines. An array of objects (class "java.util.Vector") containing 25, 50, 75 and 100 "Double" numbers respectively was passed as a parameter in the Mobile Agent, RMI and CORBA systems.

The total time required to complete the objects transfer, for each of the four different solutions described above, is reported in Figure 6, which depicts the resulting measurements in the form of statistical boxes.

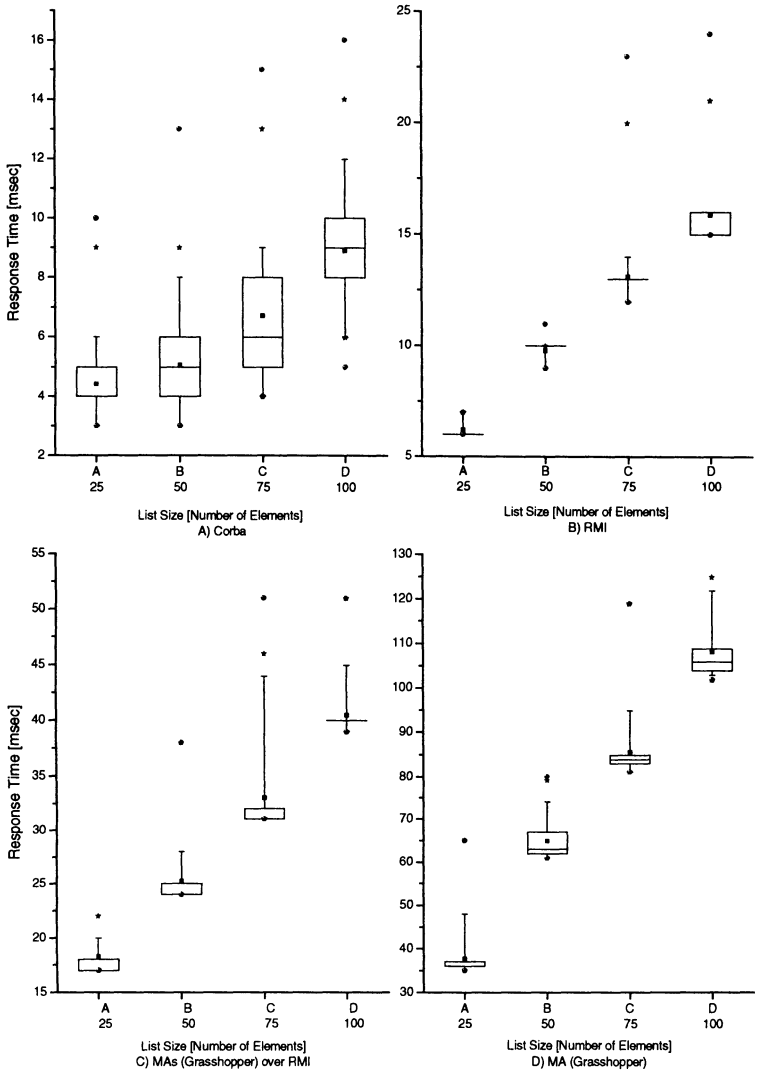

Figure 6 – Statistical Box Charts showing response times for each of the four experimented cases. The boxes include the 25-75% boundaries, the mean values (a black square) and the median values (a line). The 5-95% range boundaries are delimited by whiskers. The outliers are depicted with black circles and stars.

Figure 7 combines the same results in a single graph depicting only mean values and best linear fit, for an easier comparison.

Some important results can be drawn from Figure 6 and Figure 7. First, there is a significant performance penalty to pay for remote method invocations in the context of a mobile agent platform compared to Java-RMI and CORBA. Second, Grasshopper performs much better over RMI in comparison to the default proprietary protocol. The Grasshopper approach is at least three times slower than the Java-RMI and CORBA ones. Finally, by observing the difference in slopes from Figure 7, we can conclude that these two former approaches, along with the MA over RMI implementation, tend to scale much better than the plain MA approach.

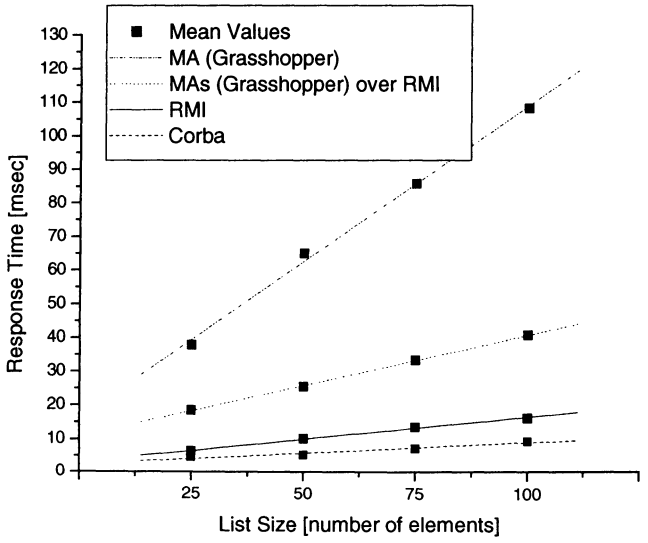

Figure 7 – Mean values and best linear fit of response times.

An additional performance overhead in Grasshopper is the initial time for the performance monitor mobile agent to migrate to the network element. The mobile agent needed an average of 1505 milliseconds to migrate, a performance overhead much larger than the time required to create a performance monitor object through its factory in the static RMI/CORBA approaches, which is less than 15 msec. In other words there is additional overhead of two orders of magnitude. In cases of constrained mobility, which is the approach used in this paper, this is a one-off overhead and can be tolerated. On the other hand, this measurement shows that agent mobility has relatively high performance overheads and this should be born in mind when designing systems exhibiting full mobility.

5.3 Total Packet Sizes

We also measured the packet sizes in all four cases. An array of objects (class *java.util.Vector*) containing 25, 50, 75 and 100 *double* numbers respectively was remotely sent using remote invocations in the Mobile Agent, RMI and CORBA systems. Each time, the payload of the TCP packets was measured. A graph of the results gathered can be seen in Figure 8. It is interesting to observe that Grasshopper configured with its proprietary protocol incurs levels of traffic comparable to those incurred by the two distributed objects systems. When Grasshopper was configured to operate over Java-RMI's JRMP it clearly incurred the biggest amount of traffic.

We also measured the packet overhead of migrating the performance monitor mobile agent to the network element. The required data was 2854 bytes, which again is much higher than the amount of data required to create a performance monitor object through its factory in the static approaches, which is around 500 bytes. This again will incur a substantial traffic overhead in full mobility environments, but can be tolerated in the case of constrained mobility.

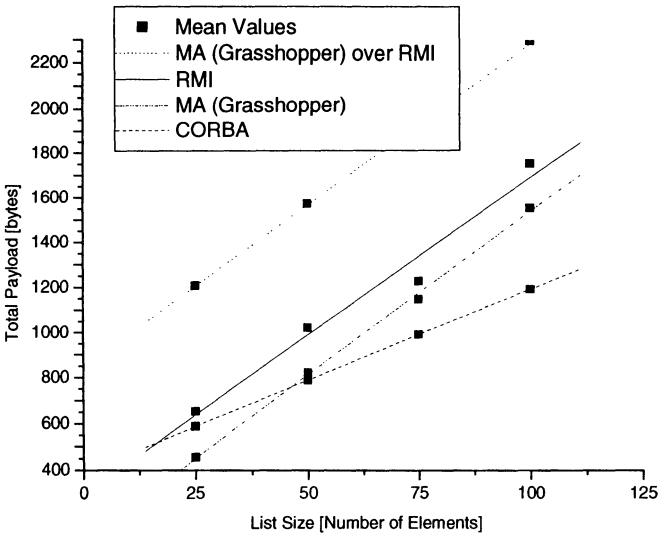

Figure 8 – Mean and best linear fit of total incurred TCP payloads, measured as the sum of all the bytes incurred in the network to complete the given task.

6. SUMMARY AND CONCLUSIONS

In this paper we present three different models of mobility that can be used in management applications. We also describe a performance monitoring system that uses the model of constrained mobility, and use it to evaluate the performance overheads of an agent based system compared with distributed ones based on Java-RMI and CORBA.

Constrained mobility involves the migration of an agent to a remote machine, where it executes a task and terminates upon completion. This is a particularly suitable model for tasks requiring a long period of time to complete. Also in scenarios where information intended for off-line analysis is collected by the agent in the remote machine. Finally programmability of network elements can be achieved in a way that the functionality at the network element level can be extended or customised, as we also demonstrated through the description of our agent-based performance monitoring system.

Weak mobility involves the migration of a mobile agent to a number of machines without preserving information gathered from previous visits. This is a suitable model for performing a short-term task repetitively in a number of machines. Also in scenarios where information intended for on-line analysis is collected by the agent in the remote machine.

Strong mobility involves the migration of a mobile agent to a number of machines while it preserves its state formed during previous visits. It is best suited for scenarios where the information collected from previous visits can affect the current or future behaviour of the agent. The task the agent has to complete in every machine should be a short term one, and therefore this model can be applied when information is collected for on-line analysis.

The nature of mobile agents does not allow a general mobility model of deployment in management applications. A suitable model can be selected by examining the requirements of a specific application.

In our performance monitoring system, mobile agents are created at the network management level according to user requests and then migrate to network elements to perform monitoring functions in a local manner. The behaviour of the monitoring algorithms can be customized, enabling dynamic programmable functionality to be provided directly in the managed network elements.

One of the key targets in embarking in this exercise was to evaluate the use of mobile agent technology in comparison to static object approaches in network management environments. The design and implementation presented in section 4 show that mobile agent platforms exhibit the same programmability characteristics to static object platforms. In addition, both remote method invocations and local invocations are possible. The same object-oriented principles and similar Application Programming Interfaces (APIs) can be used in mobile agent environments. A key advantage of mobile

agents is the provision of dynamic services in network elements that have not been pre-programmed with such facilities. The customisation of mobile agent behaviour can provide a powerful mechanism for "intelligence on demand".

On the other hand, while design and programmability aspects are similar to static object approaches, there is a performance overhead to pay when using mobile agents. Remote method invocations are at least three times slower than those in Java-RMI / CORBA and this difference could be more pronounced when comparing performance to the protocol-based OSI-SM and SNMP approaches. In addition, agent migration incurs a substantial overhead in terms of both latency and required data to be transported across the network. This is less of an issue in constrained mobility environments but could lead to performance and scalability problems in environments where a large number of mobile agents migrate relatively often – i.e., in weak and strong mobility systems.

While initially mobile agent frameworks were thought as rivals to static distributed object frameworks, a view also stated in [7], the two approaches need to coexist. Static object approaches can provide superior performance characteristics. Real synergy could thus be achieved if stationary agents could be provided using static objects, with method invocations being possible between mobile agents and static objects in both directions. Such an environment would combine the best of both worlds. We are currently working on enhancing Java-RMI with an environment supporting constrained mobility.

ACKNOWLEDGMENTS

This work was partly undertaken in the context of the ACTS MIAMI project, which is funded by the Commission of the European Union.

REFERENCES

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

[2] ITU-T Rec. X.701, Information Technology - Open Systems Interconnection, *Systems Management Overview*, 1992.

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

[4] Y. Yemini, G. Goldszmidt, S. Yemini, *Network Management by Delegation*, in Integrated Network Management II, Krishnan, Zimmer, eds., pp. 95-107, Elsevier, 1991.

[5] N. Vassila, G. Pavlou, G. Knight, *Active Objects in TMN*, in Integrated Network Management V, Lazar, Saracco, Stadler, eds., pp. 139-150, Chapman & Hall, 1997.

[6] M. Baldi, S. Gai, G.P. Picco, *Exploiting Code Mobility in Decentralised and Flexible Network Management*, Proc. of the 1st International Workshop on Mobile Agents, Berlin, Germany, April 1997.

[7] Breugst, M., Magedanz, T., *Mobile Agents - Enabling Technology for Active Intelligent Network Implementation*, IEEE Network, Vol. 12, No. 3, May/June 1998.

[8] A. Bieszczad, B. Pagurek, T. White, *Mobile Agents for Network Management*, IEEE Communications Surveys, Vol. 1, No. 1, http://www.comsoc.org/pubs/ surveys, 4Q1998.

[9] Mobile Intelligent Agents for the Management of the Information Infrastructure (MIAMI) ACTS project, project page: http://www.fokus.gmd.de/research/cc/ima/ miami/, page at Univ. of Surrey: http://www.ee.surrey.ac.uk/CCSR/ACTS/Miami/

[10] ITU-T Rec. M.3010, *Principles for a Telecommunications Management Network (TMN)*, Study Group IV,1996.

[11] ITU-T Rec. X.739, Information Technology - Open Systems Interconnection, *Systems Management Functions - Metric Objects and Attributes*, 1992.

[12] ITU-T Rec. X.738, Information Technology - Open Systems Interconnection, *Systems Management Functions - Summarization Function*, 1993.

[13] S. Walbusser, *Remote Network Monitoring (RMON) Management Information Base*, IETF RFC 1271, 1991.

[14] 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.

[15] Foundation for Intelligent Physical Agents, web page: http://www.fipa.org/.

[16] Object Management Group, *Mobile Agent System Interoperability Facilities Specification*, orbos/97-10-05, 1997, ftp://ftp.omg.org/pub/docs/orbos/97-10-05.pdf

[17] The Grasshopper Agent Platform http://www.ikv.de/products/grasshopper/index.html.

[18] D. Griffin, G. Pavlou, P. Georgatsos, *Providing Customisable Network Management Services Through Mobile Agents*, Proc. of the 7th International Conference on Intelligence in Services and Networks (IS&N'00), Athens, Greece, February 2000.

[19] M. Wooldridge, N.R. Jennings, *Intelligent Agents: Theory and Practice*, The Knowledge Engineering Review, Vol.10, N.2, pp.115-152, 1995.

[20] G. Knight, R. Hazemi, *Mobile Agents based Management in the INSERT Project*, Journal of Network and Systems Management, Vol.7, N.3, September 1999.

[21] J.W. Stamos, D.K. Gifford, *Remote Evaluation*, ACM Transactions on Programming Languages and Systems. 12(4), pp.537-565, October 1990.