Hierarchical Network Management: A Scalable and Dynamic Mobile Agent-Based Approach

Damianos Gavalas[†], Dominic Greenwood[*], Mohammed Ghanbari[‡], Mike O'Mahony[‡]

[†]Dimokratias 54,
Neo Psychiko, 15451,
Athens, Greece
E-mail: gavalas_damianos@yahoo.com

[*]Network Agent Research
Fujitsu Laboratories of America, Inc.
595 Lawrence Expressway, Sunnyvale, 94086, CA, USA.
E-mail: d.greenwood@fla.fujitsu.com

[‡]Communication Networks Research Group,
Electronic Systems Engineering Department,
University of Essex, Colchester, CO4 3SQ, U.K.
E-mail: {ghan, mikej}@essex.ac.uk

Abstract

Several distributed management architectures, incorporating mobile agent technology, have been recently proposed to answer the scalability limitations of centralised models and the flexibility problems of static hierarchical frameworks. Yet, although agent-based management frameworks have recently started evolving from the early 'flat' models to hierarchical structures, they cannot efficiently cope with the dynamically changing traffic and topological characteristics of modern networks. This is mainly due to the limited use of agent mobility (employed either through mid-level manager entities or between static mid-level managers and managed devices) and lack of appropriate policies enabling automatic calibration of the management system based on network conditions. This paper presents a hierarchical agent-based infrastructure, suitable for the management of large-scale enterprise networks that addresses these issues. The transition to hierarchical agent-based management is achieved through a mid-level manager that being a mobile agent itself, operates at an intermediary level between the manager and the legacy systems and takes full control of managing a given network segment. These entities make the system more adaptive to changing networking conditions, while localising the traffic associated with bandwidth-intensive monitoring applications. A quantitative evaluation, in terms of the overall management cost, confirms that this architecture outperforms both centralised approaches and mobile agent-based 'flat' management models.

Keywords: Distributed Hierarchical Management, Mobile Agents, SNMP, Adaptive.

1. Introduction

The increasing complexity of networks has motivated the evolution of existing management models. These models traditionally adopt a centralised, Client/Server (CS) approach wherein the functionality of both clients (managers) and distributed servers (agents) is rigidly defined at design time. Concerning formal standardisation approaches in Network Management (NM), the scene has been dominated, during the past decade, by the IETF Simple Network Management Protocol (SNMP) [23] and the OSI Common Management Information Protocol (CMIP) [11]. Within these protocols, physical resources are represented by managed objects. Collections of managed objects are grouped into tree-structured Management Information Bases (MIB). Both SNMP and CMIP follow the CS paradigm, typically associated with massive transfers of management data, which cause considerable strain on network throughput and processing bottlenecks at the manager host.

These problems have motivated a trend towards distributed management intelligence that represents a rational approach to overcome the limitations of centralised NM. As a result, several hierarchical/distributed management frameworks have been proposed both by researchers and standardisation bodies [10][13][22]. However, these models, as discussed in Section 2, are typically identified by static management components that cannot adapt to the evolving nature of today's networks, with rapidly changing traffic patterns and topology structures.

Lately, the Mobile Agent (MA) paradigm has emerged within the distributed computing field. The term MA refers to autonomous programs with the ability to move from host to host to resume or restart their execution and act on behalf of users towards the completion of a given task [1]. One of the most popular topics in MA research community has been distributed NM, wherein MAs have been proposed as a means to balance the burden associated with the processing of management data and decrease the traffic associated with their transfers (data can be filtered at the source). The majority of MA-based management platforms [3][19][24] are still based on 'flat' architectures (i.e. they do not take into account the hierarchical structure of modern networks), which only partially solve the scalability limitations of centralised architectures. This problem has been addressed to some extent by frameworks that propose a domain-based approach [5][8][20] with further improvements demonstrated by several hierarchical models [14][21][26]. However, even hierarchical architectures do not fully exploit the benefits brought about by agent mobility and, as a result, they cannot answer the flexibility limitations of proprietary management platforms. In particular, two approaches have been presented so far: (a) Use of static mid-level managers relying on MAs for the network monitoring process [14][21]; (b) Use of mobile mid-level managers either performing decentralised SNMP management [26] or integrating distributed management frameworks under standardisation [17]. Still, even the latter approach lacks clearly defined mechanisms for achieving automatic adaptation of the management system to changing network configurations, i.e. mid-level managers do not normally change the location where they execute. In addition, the use of centralised management, even between the middle and bottom levels of the hierarchy, may cause scalability problems.

This work aspires to address these issues through introducing a highly scalable and adaptive hierarchical MAF tailored to distributed NM applications. Such a model presupposes the presence of a novel management element, termed Mobile Distributed Manager (MDM), operating at an intermediary level between the manager and the stationary agents. MDMs are essentially MAs, which take full control of managing a specific network domain and localise the associated management traffic (i.e. expensive, low-bandwidth WAN links are not heavily used). Apart from the fact that management functionality may be added/configured at runtime, this architecture can also dynamically adapt to fluctuating networking conditions. Namely, an MDM entity may be assigned to / removed from a network segment to reflect a change on network traffic patterns, or move to the least loaded host in order to optimise the usage of local resources. The system's scalability is further improved by assigning monitoring tasks to MAs (launched and controlled by the MDM) capable of filtering collected data locally. MDMs are deployed to remote subnets according to policies defined by the administrator. The advantage of our proposed architecture over centralised management and 'flat' MA-based models is verified by an analytical quantitative evaluation. It is emphasised that the design ideas introduced in this paper have been already implemented; our prototype has been tested on realistic topology and application scenarios, with several measurements recorded and discussed herein.

The remainder of the paper is organised as follows: Section 2 comprises an overview of the formal and research approaches in the area of distributed NM. Section 3 reviews several MAFs proposed to distributed NM applications, whilst Section 4 explains the rationale behind our chosen hierarchical approach. Section 5 provides an overview of the MAF used to support this work. Section 6 discusses the implementation details of the introduced architecture with a quantitative evaluation of the bandwidth usage given in Section 7 and empirical results presented in Section 8. Finally, conclusions are drawn in Section 9.

2. Review of Hierarchical Management Approaches

Large enterprise networks span application, organisational and geographical boundaries. In order to cope sufficiently with the unpredictable growth of the number of network devices, structuring networks in logical hierarchies is being employed as a design and deployment principle. Any of the following or a combination of these partitioning criteria can be used: (a) geographical subdivisions; (b) administrative subdivisions; (c) grouping based on different access privileges and security policies; (d) performance-driven network partitioning between multiple management servers. The design of such networks' management systems must consider that the resultant network segments may be better managed as logical, hierarchically structured domains. A brief review of standardisation and research approaches to hierarchical management follows.

A first approach to decentralisation was the introduction of RMON (Remote MONitoring) [25] that facilitates the collection of traffic-oriented statistics by monitoring devices (probes), which provide detailed information concerning traffic activity within their local domain (e.g. an Ethernet segment).

Management distribution requirements have not been adequately addressed in the early versions of SNMP. The MIBs adding distribution support to SNMPv3 [23] were issued only in 1999. In particular, the DISMAN (DIStributed MANagement) Working Group of the IETF was chartered to define an architecture where a main manager can delegate control above several distributed management stations. Among others, the DISMAN framework provides mechanisms for distributing scripts which perform arbitrary management tasks to remote devices that implement the Script MIB [13].

SNMP Research has conducted similar work proposing the "Middle-Level Manager" [22], a dual-role entity that plays the role of the agent when managers request information, while acting as a manager for the agents located within its domain. When the MLM is placed across a WAN link, remotely from the manager platform, it obviates the necessity of performing SNMP polling. This reduces the polling traffic on the WAN link, thereby achieving significant cost savings.

The concept of management distribution has been pushed much further by Management by Delegation (MbD) [10]. MbD has been the first attempt to address decentralisation and automation of management tasks by dynamically delegating management functions to stationary agents ("*elastic processes*").

The main weakness of the aforementioned standardisation and research approaches to hierarchical management is their inflexible and static definition. There are two parts to building such management hierarchies: (i) assigning roles (e.g. "Mid-Level Manager") to the members of the hierarchy, and (ii) assigning members to specific physical locations that will function under the supervision of higher-level members. This is feasible if the network is moderately small and/or not very dynamic, and if a single administrator is the only user and specifier of the management policies. However, it is not in step with the dynamically evolving topological and traffic characteristics of large-scale enterprise networks.

In addition, the distribution of aggregation and filtering computations in these approaches is manual and static. The administrator of such a system is required to know the managed network well enough to build a hierarchy of distributed servers that will accommodate, to a certain degree, all desirable computations that might be performed on NM data. Also, the MLMs, SubManagers, etc, may suffer from overloads while executing their tasks, either due to limited capacity or insufficient computing power of the hosting processors.

Nevertheless, all these formal and research efforts point in the right direction to handle the present complexity of NM: "Divide & Conquer". Considering the technological issues associated with distributed management, we discern a field that the mobile agent paradigm can revolutionise by enriching system functionality.

3. Distributed Management based on Mobile Agents: Research Approaches

The use of mobile code has attracted tremendous attention during the last few years. MAs offer a new powerful abstraction for distributed computing, answering many of the flexibility and scalability problems of traditional centralised archetypes. Regarding distributed NM area, MAs can provide all the functionality offered by static delegation agents, having the additional benefit of mobility. In that sense, MAs can be regarded as a 'superset' of MbD agents, leading to more efficient use of computing resources on the managed entities, as management functions are executed only so long as the MAs reside and are active on the NEs [9]. On the other hand, the use of MAs imposes extra strain on the physical resources of remote hosts, brings about performance concerns and introduces potential security threats [6]. The trade-offs of using MAs for management applications as opposed to simply downloading code to remote devices are discussed in [1].

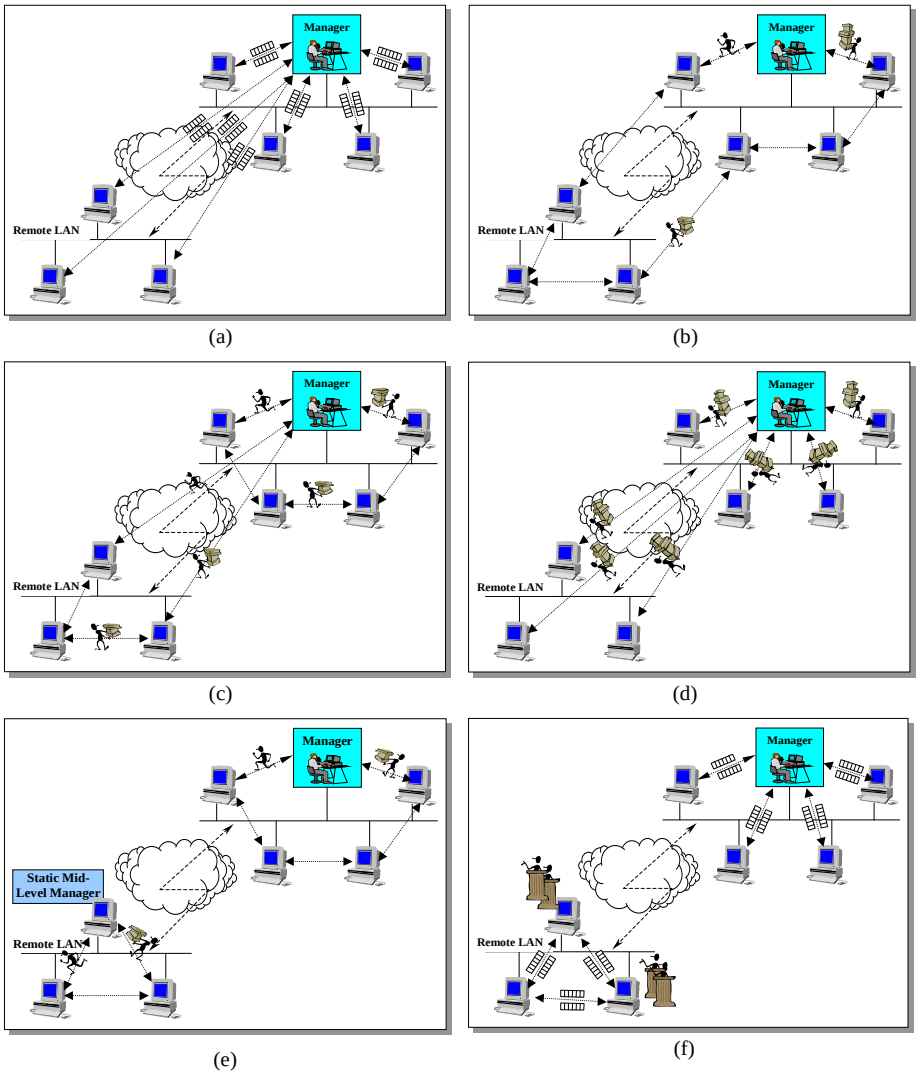


Figure 1. Centralised and distributed (MA-based) approaches to Network Management

Most of the MA-based frameworks proposed for network monitoring applications [3][19][24], assume a 'flat' network structure, i.e. a single MA is launched from the manager platform and sequentially visits all the managed NEs, regardless from the underlying topology (Figure 1b). One of the first prototypes proposed for NM has been presented in [24], where issues related with access provisioning to managed resources and communication between MAs are discussed. The framework described in [3] (extends previous work introduced in [5]]) enforces a strict security scheme and addresses interoperability issues through providing compliance with CORBA [4]. Pualiafito et al. [19] introduce the Mobile Agent Platform (MAP), used for monitoring the systems state by calculating aggregation functions combining

several MIB values (*health* functions). Nicklisch et al. [16] present the INCA architecture that supports agent prioritisation for timely execution of critical tasks and identifies multiple agent code transfer schemes.

However, although relaxing the network from a flood of request/response SNMP messages (see Figure 1a), such an approach brings about scalability issues, especially when frequent polling is required. That is, in large networks the round-trip delay for the MA will greatly increase, whilst the network overhead may overcome that of the centralised paradigm (the MA size will grow after visiting each of the nodes included into its itinerary [2]). The situation seriously deteriorates when considering management of remote LANs, connected to the backbone network through low-bandwidth, expensive WAN links. In this case, frequent MA transfers are likely to create bottlenecks and considerably increase the management cost.

A first step to address this problem has been realised through a "*segmentation*" approach [5][8], whereby the network is partitioned into several domains and a single MA object is assigned to each of them (see Figure 1c). The model presented in [5] also allows for several management levels through organising a hierarchy of the *domains* abstraction. The "segmentation" approach introduces a high degree of parallelism in the data collection process, thereby reducing the overall response time. That has been verified by simulation results presented in [20].

Alternatively, when acquired data are to be analysed off-line, the "*broadcast*" approach [8] may be used: an MA object is broadcasted to all managed devices and remains there for a number of PIs (defined by the administrator) collecting an equal number of samples before returning to the manager (see Figure 1d). It is noted that although improving the management system scalability, "segmentation" and "broadcast" schemes cannot cope efficiently with the management of remote LANs, since when data acquisition is required in real-time, the MA transfers though the WAN link are not decreased. A comparison of Figures 1b and 1c confirms that in both flat and "segmentation" polling, MA objects traverse the WAN link twice per PI, in order to visit the remote site and bring back the collected results. On the other hand, "broadcast" polling is unsuitable for time critical applications [8].

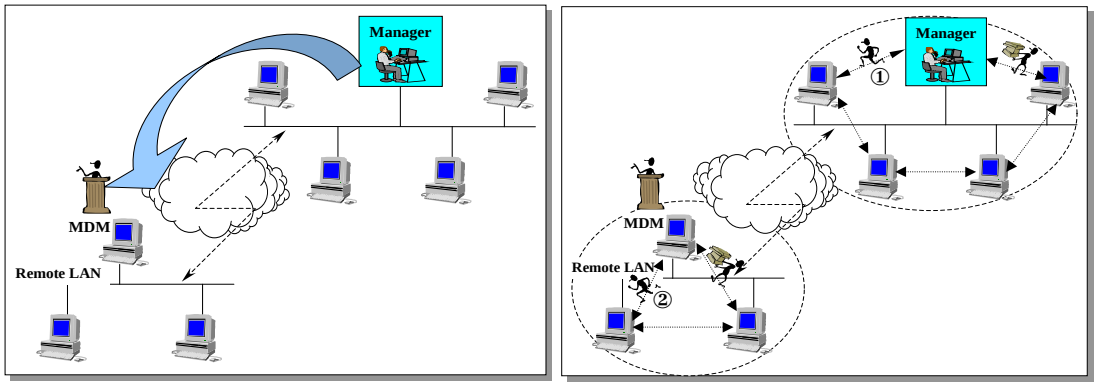
The scalability problem is more adequately addressed by hierarchical models that have recently started coming into the picture. In particular, Liotta et al. [14] have conducted an interesting study of an MA-based management architecture adopting a multi-level approach enabled by static "*Middle Managers*" able to launch MAs; interesting cost functions corresponding to various MA configurations are also proposed. Sahai & Morin [21] introduce the concept of "*Mobile Network Manager*" (MNM), an application that may execute on portable computers and assist the administrator to remotely control his/her managed network, through launching MAs to carry out distributed management tasks. The heavier part of the management functionality is integrated into a static "*management server*"; the framework is enriched with features such as fault tolerance, adaptability of MAs to changes in the environment (they are capable of detecting the disappearance of an agent server), etc. In [17], Oliveira and Lopes propose the integration of the IETF's DISMAN framework into their MA-based NM infrastructure. However, the described "*Mobile Disman*" architecture is heavyweight (it consists of many resource-demanding components) and would certainly increase the requirements on system resources; it must also be considered that DISMAN is still very much under development. In addition, [14] and [17] are not supplemented by prototype implementations. The framework presented in [26] addresses scalability issues by delegating NM tasks to MAs that migrate to remote domains where they act as local managers, performing SNMP operations. Several interesting applications are proposed, including evaluation of health functions, termination of mis-behaving processes to free-up system resources, etc. Nevertheless, since each MA corresponds to a single management task, the introduction of additional services will trigger the deployment of an equal number of independent MAs that will not necessarily execute on the same host (see Figure 1f). This approach though, is not in line with the concept of a *compact* mid-level manager entity responsible for *all* the decentralised operations performed in its domain, which in our opinion offers better grouping, organisation and control over distributed NM tasks.

In the aforementioned works, agent mobility is either exploited between the middle and bottom levels of the hierarchy to carry out the network monitoring process [14][21] (see Figure 1e), or integrated within mid-level managers that rely on standardised frameworks for accessing the legacy systems [17][26]. In the former case, the problems of static hierarchical frameworks listed in the preceding section still remain unsolved. Yet, even the latter approaches lack clearly defined mechanisms for achieving automatic adaptation of the management system to changing network configurations, i.e. mid-level managers do not normally change the location where they execute [26]. In addition, critical issues such as well-defined criteria for segmenting the network into management domains, explicit determination of the domain boundaries or strategies for assigning mid-level managers to these domains, are not elucidated.

A direct application of these approaches in distributed systems management is far from straightforward. Although the problems that need to be solved have been identified, the rules that define the mid-level entities deployment strategy, i.e. questions concerning *when* and *where* to deploy, remove or change the location of mobile mid-level managers still remain open. In addition, scalability issues may arise as a result of using centralised management, even between mid-level managers, since the ability of MAs to filter management at the source in a variety of applications [9] is not fully utilised. Hence, the deployment of a highly adaptive hierarchically structured MA-based management model seems a rational approach to address these issues as well as to overcome the limitations of statically configured NM frameworks presented in the preceding section.

4. Hierarchical, Mobile Agent-based Network Management

In search of more flexible solutions, this work aspires to push the concept of MA-based Hierarchical/Distributed Management much further. Specifically, we introduce the novel concept of Mobile Distributed Manager (MDM), referring to a management component that operates at an intermediary level between the manager and management agent end points. MDM entities are essentially MAs that undertake the full responsibility of managing a network domain, when certain criteria (determined by the administrator) are satisfied. Upon being assigned to a domain, the MDM migrates to a host running in that domain (Figure 2a) and takes over the management of local NEs from the central manager.



(a) (b)

Figure 2. Hierarchical MA-based management

As a result, the traffic related to the management of that domain becomes localised, as the MDM is able to dispatch and receive MAs to collect NM data from the local hosts (Figure 2b), or even execute centralised management operations on them. The MDM continues to perform its tasks without the manager's intervention, even if the interconnecting link fails. A first-line response is also given to tackle trivial faults/alarms, with the manager being notified only in case of a complex problem or an emergency situation. In performance management applications, only aggregated values and statistics are sent to the manager at regular intervals,

thereby diminishing the amount of data transferred through the WAN link. The duration of these intervals is application-dependent and determined by the administrator.

The decision concerning the selection of the host where the MDM will carry out its management tasks from, will be discussed in a later section. It is noted that the management domain assigned to an MDM entity may be confined to a single network segment or expand to a larger set of hosts.

The mobility feature of MDMs allows the management system to adapt dynamically to a fluctuating environment, optimising the use of network resources. Management functionality may be downloaded at runtime, while this architecture can also dynamically adapt to changing networking conditions. Namely, an MDM entity can be deployed to / removed from a network segment in response to a change in network traffic distribution, or move to the least loaded host to minimise the usage of local resources.

The application chosen as a case study for our test-bed is 'untargeted' network monitoring over a large set of NEs with periodic data collection. Fault management applications have not been examined. However, given that our platform co-operates with a fault management system, it can limit the data collection process to a subset of NEs wherein a potential miss-behaviour has been identified, i.e. MAs acting as data collectors can target a restricted set of NEs, until the fault is isolated.

In summary, our proposed architecture meets the following design requirements:

- *Integration with existing management standards*: Our architecture encompasses the dominant NM framework of the Internet world, i.e. SNMP. Due to its huge installation base, integration with SNMP was considered of vital importance to maintain compliance with legacy management systems.
- *Mobility support*: The ability of management components to move from host to host offers a powerful abstraction that should not be disregarded when building distributed applications. The architecture should therefore provide a set of services to allow the migration and cloning of management agents, regardless of the underlying management models.
- *Modularity*: The management community has not yet reached a consensus on the choice of a NM model, resulting in several different solutions available, either proprietary or standard. Maintaining architectures with intrinsic modularity eases the interoperability between different vendor models. In addition, modularity makes feasible the addition of new services or the modification of existing ones.
- *Fault-tolerance and robustness*: MDM entities should be able to detect and tolerate failures on the inter-connecting link between their local subnet and the manager's site or on the manager platform itself; it is essential that the management process is not disrupted should such failures are detected.
- *Load balancing*: The total workload should be equally distributed among the various processors of the underlying subsystems. MAs can take full advantage of the increasing processing capability of network devices to achieve management intelligence distribution, however that should not lead to exhaustive consumption of local resources. In particular, the stationary 'agencies', interfacing between incoming MAs and legacy systems, should have a minimal footprint on local devices, whilst MDMs should be designed as light-weight as possible, i.e. they should be equipped with basic management functionality.
- *Minimal intrusiveness*: MDMs should be deployed at specific hosts so as to minimise their intrusiveness in terms of the effect of management-related traffic on other applications and the additional processing burden placed upon host processors. In addition, MAs itineraries should be decided in an intelligent, non-random manner, aiming at maximising efficiency and reducing network overhead.
- *Dynamic adaptation*: Topological and traffic characteristics of today's networks are rapidly changing. A hierarchical NM system should therefore be flexible enough to adapt to those changes. Hence, the location where MDMs run is not fixed, neither is the set of hosts under their control. MDMs can be transparently sent to a domain when the associated cost savings are considerable or removed when their existence is no longer

necessary. They can also autonomously decide to move within their domain when the host processor is overloaded and continue their operation on the least loaded node.

- *Ease in introducing new services*: It is essential to provide an open architecture in which the administrator can easily add new services or modify existing ones at runtime. In our framework, there is a direct mapping of management functions to certain MA objects. The effortless introduction of new services is accomplished through a graphical tool that automates MA code generation and eases the deployment of new MAs that carry out specific management operations.

5. The Mobile Agent Framework

The MAF that comprises the core of the hierarchical infrastructure has been entirely developed in Java chosen due to its inherent portability, rich class hierarchy and dynamic class loading capability.

Our framework consists of *three* major components [7], illustrated in Figure 3:

(I) Manager Application: The manager application, equipped with a browser style Graphical User Interface (GUI), co-ordinates monitoring and control policies related to the NEs. Active agent servers are automatically *discovered* by the manager, which maintains and dynamically updates a 'discovered list'.

(II) Mobile Agent Server (MAS): The interface between visiting MAs and legacy management systems is achieved through MAS modules, installed on every managed device. The MAS logically resides above a standard SNMP agent, creating an efficient run-time environment for receiving, instantiating, executing, and dispatching MA objects.

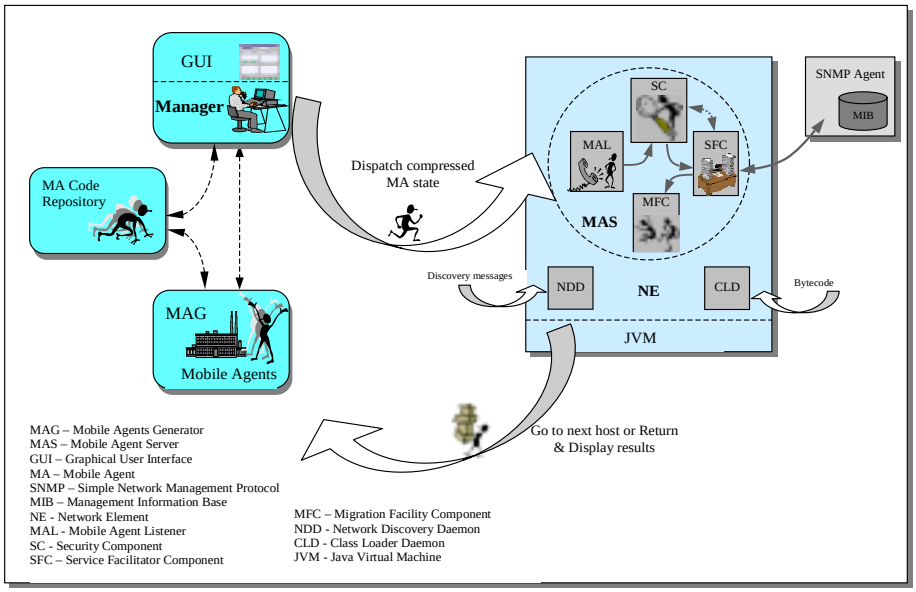


Figure 3. The Mobile Agents-based NM Infrastructure

The MAS also provides requested management information to incoming MAs and protects the host system against external attack. The MAS composes four primary components: (a) Mobile Agent Listener, (b) Security Component, (c) Service Facility Component, and (d) Migration Facility Component. Special focus has been given on the design of the Security Component in order to face the security threats represented by executing MAs [6]. Thus, in addition to the *authorisation* of the MAs requests, the RSA algorithm has been implemented to provide *authentication* of incoming MAs and *encryption* of the obtained sensitive NM data.

(III) Mobile Agents (MAs): From our perspective, MAs are Java objects with a unique ID, capable of migrating between hosts where they execute as separate threads and perform their

specific management tasks. MAs are supplied with an *itinerary table*, a *data folder* where collected management information is stored and several methods to control interaction with polled devices. As described in [7], service-oriented MAs, associated with new management tasks, may be created at runtime using the Mobile Agent Generator (MAG) graphical tool. The MAG automatically generates and compiles the code of MAs; MA classes may extend one of the provided super-classes (corresponding to general patterns of NM tasks).

As mentioned in [9], the multi-node movement of MAs can be exploited in a variety of data filtering applications. In particular, MAs may: (i) aggregate several MIB values into more meaningful values, (ii) efficiently acquire atomic snapshots of SNMP tables, and (iii) filter tables' contents by applying complex filtering expressions thereby keeping only the values that meet pre-specified criteria.

6. Implementation Details

6.1. Topology Map of Active Devices

An important element of our framework is the *topology map*, a graphical component of the manager application, used to view the devices with currently active MAS servers. This component not only shows the discovered active devices, but also the underlying network topology, namely the subnetworks where these devices are physically connected as well as how these subnetworks are interconnected.

In terms of implementation, the topology map is internally represented by a tree structure (termed the "*topology tree*"), where each of the tree nodes corresponds to a specific subnetwork. The node representing the manager's location is the root of the topology tree (see Figure 4).

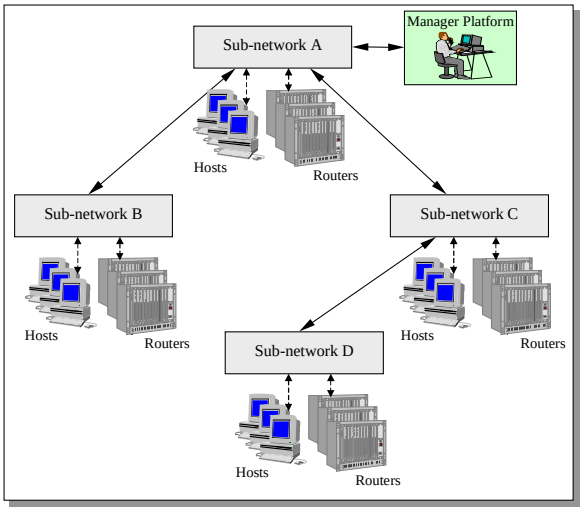


Figure 4. The topology tree structure

Each of the tree nodes consists of the following attributes:

- the subnetwork's name;
- the names of hosts and routers physically connected to this subnetwork;
- a flag indicating the presence of an active MDM on this subnetwork;
- the number n_l of local active hosts on this subnetwork;
- the number n_s of active hosts on the subnetwork's 'subtree' (the term subtree here denotes the set of subnetworks located in hierarchically lower levels in the topology tree, including the present subnetwork itself), hence $n_s \geq n_l$;
- a pointer to the upper level tree node;
- pointers to the next level nodes;
- a list of graphical components, each corresponding to a specific host, that will be made visible upon discovering an active MAS entity on that host.

For instance, the number of active hosts in the subtree of Subnetwork A (in Figure 4) will be:

$$n_{s,subA} = n_{l,subA} + n_{l,subB} + n_{l,subC} + n_{l,subD} \qquad (6\text{-}1)$$

All the information related to the managed network described above, is given to the manager application upon its initialisation, through parsing a text file (*"network configuration"* file). That file does not include, of course, the activity status information, which is automatically discovered as described in the preceding section. For each file entry, a new subnetwork node is created and inserted into the topology tree. In particular, its 'parent' (upper-level) subnetwork is located and then the next-level pointer of the parent node as well as the upper-level pointer of the inserted node are updated.

As described below, the topology tree plays a crucial role when the manager application needs to make a decision on which subnetworks require the deployment of an MDM entity.

6.2. MDMs Deployment Policies

A key characteristic of this work is the dynamic adaptation of our architecture to changes in the managed network. The structure of the proposed model is not rigidly designed, as MDMs may be dynamically deployed to specific network domains, given that certain requirements are met. Specifically, the administrator may explicitly set (through the GUI shown in Figure 5) the policies that define the hierarchical NM system operation, i.e. specify the criteria that should be satisfied for deploying an MDM to a network segment.

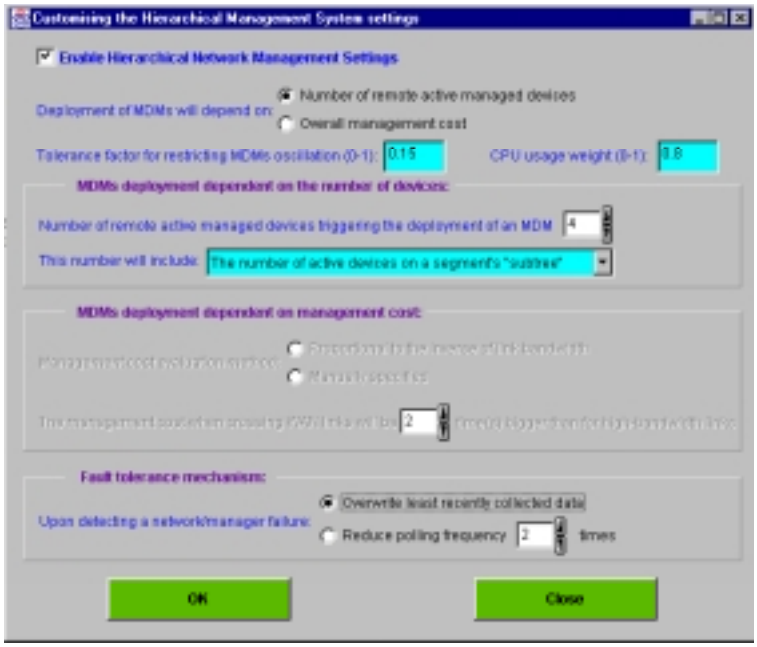


Figure 5. Graphical User Interface for customising the hierarchical NM system policies

In general, the deployment of MDMs may conform to either of the two following policies:

- **Policy 1**: the population of remotely active managed devices.
- **Policy 2**: the overall cost involved with the management of a remote set of devices.

When applying *Policy 1*, the administrator specifies the number of remote managed NEs that will justify the deployment of an MDM to a particular network segment. This number may either denote n_l or n_s. If, for instance, the specified number N denotes the population of the examined subnetwork's local devices n_l, an MDM will be deployed to every network segment S with $n_{l,S} \geq N$; the boundaries of the domain assigned to the MDM will then be limited to that segment. If, on the other hand, N denotes the active hosts on the subnetwork's

'subtree' n_s, the domain assigned to the MDM will include all the active hosts located within the examined subnetwork's subtree, excluding the hosts already assigned to another MDM. The MDM will initially migrate to the least loaded host included into its assigned domain (this issue is discussed in detail in Section 6.4). Accordingly, when the population of NEs directly managed by an MDM exceeds a certain limit, that domain will be divided to two independent domains, with another MDM undertaking the management of the second domain.

When applying *Policy 2*, the management cost may either be: (a) proportional to the inverse of link bandwidth, or (b) manually specified.

6.3. MDMs Deployment Implementation

Upon discovering an active MAS module, the corresponding host is located through scanning the topology tree and finding the subnetwork where the host belongs, whilst the host icon is instantly made visible on the topology map. Then, the number n_l of active hosts on that subnetwork is increased by one and subsequently, through following the pointer to the upper-level nodes, all the topology tree nodes up to the root are traversed and their number n_s of subtree nodes is also updated (increased by one). A similar procedure is followed when an MAS server is being shut down.

The discovery or termination of an MAS server triggers an event at the manager host. The topology tree is then scanned with the subnetworks that meet certain requirements added to a list. In case that 'Policy 1' is employed, referring to the policies listed in the preceding section, that list will include the subnetworks with n_l or n_s (depending on whether the MDMs deployment is a function of the active devices running locally or in the whole subtree) greater than the specified constant N. If 'Policy 2' is employed, the cost corresponding to the management of each subnetwork is evaluated and the list of subnetworks created accordingly. Ultimately, an MDM will be deployed to each of the subnetworks included in the list.

Certainly, the set of management tasks already performed by the manager on these subnetworks will need to be conveyed to the MDM deployed therein. This is achieved through sending the *Polling Threads* (PT) [8] configurations along with the MDM. PTs are originally started and controlled by the manager application with each of them corresponding to a single monitoring task. Unfortunately, PTs cannot be transparently transferred along with the MDM retaining their execution state, as Java does not support threads serialisation/deserialisation. Upon its arrival at the remote subnetwork, the MDM instantiates the PTs using their configurations. The PTs will thereafter start performing their tasks without any further disruption of the management process: they launch the required number of MAs (supplied with their corresponding itinerary) and then 'sleep' for one polling interval (PI). When this period elapses the same process is repeated. Meanwhile, an MDM's listener daemon receives the MAs that return carrying their collected data.

6.4. Processing Load Balancing

Although MDMs have been designed to be as lightweight as possible, they cannot avoid consuming memory and processing resources on the NE where they execute. The framework should therefore be sufficiently flexible to allow MDMs to autonomously move to another host, when their current hosting device is overloaded, in order to provide a more balanced distribution of the overall processing load.

This is accomplished through the regular inspection of the domain's NEs, in terms of their memory and CPU utilisation: an MA object is periodically dispatched and visits all the local devices obtaining these figures before delivering the results to the MDM. Host load figures represent their average load over relatively long time windows to avoid sensitivity to sporadic utilisation peaks. If the hosting processor is seriously overloaded, compared to the neighbouring devices, the MDM will transparently move to the least loaded node. MDMs are prevented from continuously oscillating between different hosts through adjusting the value of the "*tolerance factor*", $0< t_f < 1$. Assuming that an MDM executes on a host x, with average utilisation U_x (this is a linear function of the CPU and memory usage), the MDM will

consider migrating to another host *y* only if its utilisation is $U_y < (1 - t_f) U_x$. If for instance, $t_f = 0.2$, the MDM will not migrate to *y* unless its utilisation level is at least 20% lower than that of *x*.

The MDM notifies the manager application about its new location of execution before the actual migration occurs. Should the manager attempt to contact the MDM while the latter is still moving, an exception will be thrown and a new attempt to contact the MDM will take place after a specified interval; in the meanwhile, the MDM arrives at its new hosting device and is able of receiving manager's messages.

6.5. Manager-MDMs Communication

One of our framework key advantages is that it greatly reduces the amount of information exchanged between the manager platform and the managed devices. This is due to the introduction of the intermediate management level (MDMs). However, that does not obviate the necessity for bi-directional communication between MDMs and the manager host. In particular, MDMs often need to send the manager the statistics obtained through filtering raw data collected from the local devices, inform the manager when migrating to another host, etc. In the opposite direction, the manager may request an MDM to terminate its execution or move to another domain, update a PT configuration, undertake the management responsibility for a managed NE that has just started execution on the MDM's local segment, download in runtime an additional management service, i.e. a new MA object along with its corresponding PT definition, etc. We have chosen Java RMI [12] for implementing the communication bus between the distributed MDMs and the manager host, due to its inherent simplicity and the rapid prototype development that it offers.

6.6. Fault Tolerance

MDM entities have been designed so as to tolerate failures on the interconnecting link between their local subnet and the manager's site or on the manager platform. Such failures are typically detected when MDMs attempt to deliver aggregated results to the manager. Upon detecting a failure, they continue to perform their decentralised tasks as normal, while periodically checking for the status of the link and/or the manager. As soon as the communication is restored, all management data collected in the meanwhile are returned to the manager. Certainly, in case that a large number of monitoring tasks are controlled by the MDM, a prolonged disruption of the communication flow between the MDM and the manager would result in significant growth of the MDM's size. That may in turn have a serious impact on the MDM's hosting device resources. Hence, in order to maintain control over the growth of the MDMs state, the administrator may choose (through the GUI shown in Figure 5) between the following models: As soon as an MDM detects a failure it will either: (i) overwrite the least recently collected management data by keeping only the latest acquired values, or (ii) reduce the polling frequency of the individual monitoring tasks so that the management data accumulation rate will decrease.

7. Quantitative Evaluation

A critical omission frequently noticed in research papers introducing new MA-based frameworks for NM is the analytical evaluation of the performance issues arising when implementations of the corresponding models are to be used in real networking environments. With very few exceptions (e.g. [8], [26]), in most of these works, it is just claimed that their presented prototypes perform better than other approaches, but no further proofs are provided. Reference [2] comprises an interesting theoretical investigation of the three Mobile Code paradigms (Code On Demand, Remote Evaluation and MAs). A general performance comparison among these approaches is also provided, that may serve as a reference point for related quantitative evaluations. In this section, we undertake an extensive evaluation, in

terms of bandwidth usage, tailored to our concrete implementation and utilised in Section 8 to prove the prevalence of the proposed model over centralised and flat MA-based management.

Although mobility can often be beneficial for NM, overheads induced by MAs and MDMs in particular, e.g. due to their deployment and management should be accounted for very carefully. Slightly different configurations for a set of MDMs may result in dramatically variant network loads [14]. Hence, it is crucial to define concrete cost functions estimating the corresponding overheads.

In this context, let the "cost coefficients" k_{S_i,S_j} denote the cost of sending a byte of information between arbitrarily indexed subnetworks S_i and S_j, where S_0 is the manager host location. For multi-hop connections, the cost coefficients will be equal to the summation of the individual links coefficients. In the following investigation, we make the simplifying assumption that an MDM may manage only the hosts included in a single subnetwork and not a wider set of devices.

Examining a simple performance management application, let us first evaluate the management cost imposed from SNMP-based management. If S_{req} is the average request size (at MAC layer), and polling of N devices, each for v operational variables, is applied, the wasted bandwidth for p PIs would be:

$$C_{SNMP} = \sum_{i=0}^{N} k_{S_0,S^i} * \left[\left(2 * S_{req}\right) + \left(v-1\right) * \Delta S_{req}\right] * p \qquad (7\text{-}1)$$

where every extra value included in the SNMP response packet's varbind list represents an additional overhead of ΔS_{req} bytes, on average. The index S^i represents the subnetwork including the host i.

A simple function characterising the bandwidth consumption for our hierarchical architecture, is the following:

$$C_{hier} = C_{distr} + C_{depl} + C_{pol} + C_{deliv} \qquad (7\text{-}2)$$

where the four terms represent the cost for distributing to the MAS servers the bytecode of the MA that will undertake the monitoring task, the MDMs deployment cost, the bandwidth used for the actual monitoring operation (polling) and the cost for delivering to the manager host the collected data, respectively. The first two terms are expected to dominate on the overall cost only for short-term monitoring tasks, whilst the latter two dominate for management tasks with prolonged duration.

Concerning bytecode distribution, a lightweight scheme is adopted. The majority of MAFs proposed for management applications [3][21][24], with the exception of INCA [16] and NetDoctor [26], involve transfer of both the MA's code and state on each migration. Instead, we have chosen to adopt the "*push*" scheme (defined in [16]), whereby bytecode is distributed at the MA's construction time and only the state information transferred thereafter, resulting in a much lower demand on network resources (bytecode size is typically much larger than state size [2]). The code distribution scheme proposed in [3][21][24] offers a better starting point in terms of the associated network overhead, since the bootstrapping procedure described above is not required. However, it is outperformed by the scheme adopted by our MAF, after a small number of PIs.

The introduction of MDMs reduces the code distribution cost even further: the MAs bytecode is no longer broadcasted to all managed devices, as in [7], but instead it is distributed to the active MDMs, which in turn multicast it to the local NEs. The code distribution cost is therefore given by:

$$C_{distr} = \left(k_{S_0,S_0} * N_0 + \sum_{i=0}^{M} \left[k_{S_0,S^i} + k_{S^i,S^i} * \left(N_i - 1\right) \right] \right) * C \qquad (7\text{-}3)$$

where M is the total number of active MDMs, C the compressed bytecode size and N_i the number of hosts included in subnetwork S^i.

Likewise, C_{depl} is equal to the cost of broadcasting M MDM objects to their corresponding remote domains:

$$C_{depl} = \sum_{i=0}^{M} k_{S_0,S^i} * (C_{MDM} + ST_0 + n * \overline{S}_{conf}) \tag{7-4}$$

where C_{MDM} is the MDM class size (≈ 9 Kb), ST_{i} represents the compressed state size of an MA when migrating from the i^{th} host and $\overline{S}_{conf}$ the average size of each of the n PT configurations attached to the MDM (≈ 250 bytes).

C_{pol} is defined as the summation of the cost induced for polling the NEs being directly managed by the manager host and the cost associated with polling the NEs that operate under the MDMs control, multiplied with the number of PIs:

$$C_{pol} = \left(\sum_{i=0}^{m} k_{S^i,S^{i+1}} * ST_i + \sum_{i=0}^{M} \sum_{j=0}^{N_i} k_{S^i,S^i} * ST_j \right) * p \tag{7-5}$$

Clearly, the first term of the summation will dominate on the overall polling cost if the m devices managed by the central manager platform are spread among several subnetworks. Specifically, cost coefficients $k_{S^i,S^{i+1}}$ are typically larger when an MA migrates from subnetwork S^{i} to another subnetwork $S^{\text{i+1}}$ ($S^i \neq S^{i+1}$) rather than when it moves within the same subnetwork ($S^i = S^{i+1}$). It is emphasised that MAs state size ST_{i} does not remain constant, but increases for each visited node. Thus, the polling cost highly depends on the increment rate of the MAs state size, which in turn is a function of "selectivity" σ, a metric defined in [14] as the ratio of the amount of data ultimately delivered to that acquired from each host. It is apparent that for small selectivity values (the major part of the obtained data being filtered at the source) the MAs state size will practically remain constant, otherwise the state will rapidly grow. Thus, if b bytes of information are obtained at each host, an MA's state size at its i^{th} hop is given by:

$$ST_i = ST_0 + (\sigma * b) * i \tag{7-6}$$

The last term appearing in Eq. (7-2) represents the cost associated with the delivery of the gathered data from the MDMs to the manager host:

$$C_{deliv} = \frac{D * p}{t} * \sum_{i=0}^{M} k_{S^i,S_0} \tag{7-7}$$

where t indicates (in number of PIs) how often MDMs package the computed statistics of size D and deliver them to the manager.

It needs to be emphasised that MDM functionality is not limited to simply gathering and delivering data to an upper-level manager. Although this paper concentrates on data-intensive network monitoring applications as a case study, a broad spectrum of management applications (including fault, configuration and security management) could be also performed. Upon arriving at their remote domains, MDMs may autonomously make management decisions and take actions based on the values of collected MIB values (for instance when the value of an aggregation function of several MIB objects exceeds a pre-specified threshold). These actions may include first-line support to handle trivial faults, perform simple configuration tasks, decisions to recalibrate the management system as a response to changes sensed to traffic patterns or network configuration, e.g. to share the management responsibility of its domain with another MDM, move to a nearby domain, terminate execution, etc.

8. Empirical Results

The quantitative model introduced in the preceding section has been applied to the test network shown in Figure 6, where the network domain margins are depicted by the dotted curved lines. Referring to this particular topology, we assign the cost coefficients the

following values: $k_{S_0,S_0} = k_{S_1,S_1} = k_{S_2,S_2} = 1$, $k_{S_0,S_1} = 5$ and $k_{S_0,S_2} = 50$. Coefficient values are chosen in accordance to the bandwidth available on the links they correspond to. We have also measured the following variables values: S_{req} = 90 bytes, ΔS_{req} = 17 bytes, C = 1.95 Kbytes, and ST_0 = 447 bytes. These values have been measured after testing our framework in a real network comprising Solaris and WinNT devices.

Equations (7-1) - (7-7) are applied to compare the performance of SNMP polling against that of MA-based flat and hierarchical NM in terms of the overall management cost, as shown in Figure 7, drawn on a logarithmic scale. The functions defining the cost of MA-based flat management represent special cases of those developed for hierarchical management.

We consider a data intensive application, namely polling every host for the contents of the MIB-II *interfaces* table [15]. We assume that there are two interfaces per host, i.e. 2×21 collected values per host, since each table row includes 21 columns. As described in [9], the MA objects are able of performing local filtering of the obtained data, so that only the values corresponding to the more heavily loaded interface are being encapsulated into the MA's state and returned to the manager or the MDM that originally launched the MA. This results in improved system scalability, due to the low selectivity ratio σ achieved over the acquired data. In particular, we have measured an MA state size increment of only 13 bytes for each visited host ($\sigma \times b = 13$). We also assume that management data are gathered by the MDM and delivered to the manager at regular intervals (in this case, every 10 PIs). The MDM that runs on Subnet 2 for instance, having to poll 4 devices, it gathers D = 4_devices × 13 bytes/device = 52 bytes every PI and delivers (through an RMI call) to the manager a total of 10 × 52 = 520 bytes every 10 PIs.

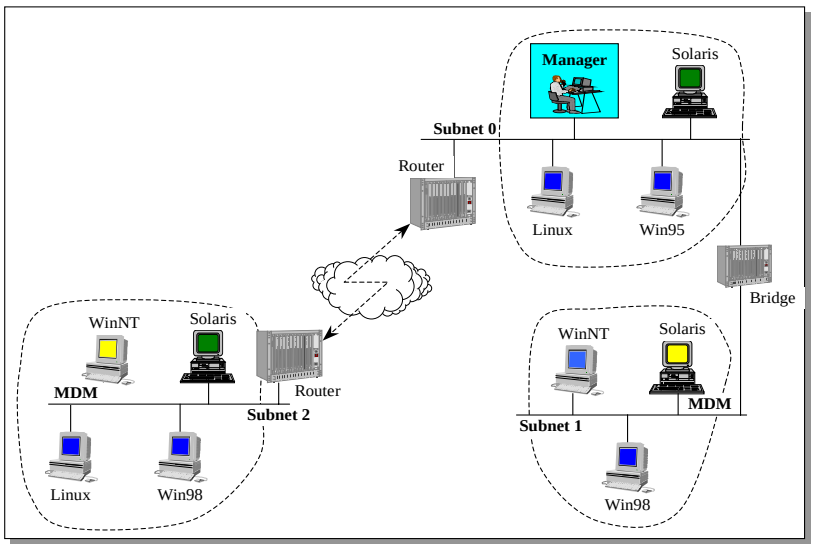


Figure 6. The test network

Clearly, the introduced hierarchical architecture gives rise to a remarkable reduction of management cost, while the cost of flat management is surpassed by that of centralised polling only after the first *16* PIs. It is also noted that the starting point for the cost induced by the hierarchical infrastructure is much lower than the equivalent of flat management, due to the adopted scalable code distribution scheme, described in the preceding section.

It should be also emphasised that for both SNMP and MA-based flat management, whenever a packet or an MA object is sent through a link, the traffic associated with the actual transfer affects both the network segments attached to that link. For instance, if an MA was to poll all the devices of Subnet 1 and then (on its i^{th} hop) move to Subnet 2 to continue its execution, the migration cost for the transfer between Subnet 1 and Subnet 2 would be: $\left(k_{S_1,S_0} + k_{S_0,S_0} + k_{S_0,S_2}\right) * ST_i$. In contrast, hierarchical MA-based management relaxes the

network from this unnecessary traffic burden, as after the MDMs deployment, the management traffic is localised thereby minimising the use of interconnecting links.

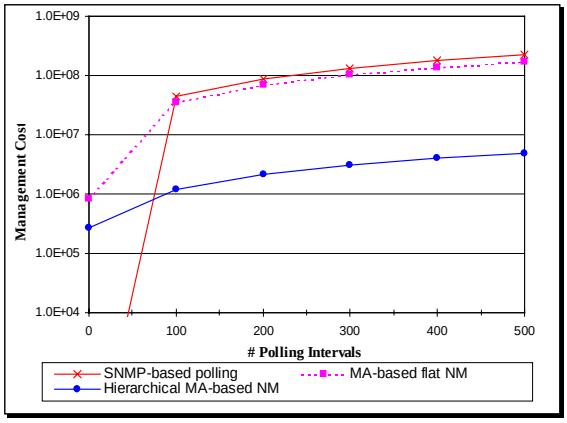


Figure 7. Management cost of hierarchical framework against SNMP and flat MA-based polling

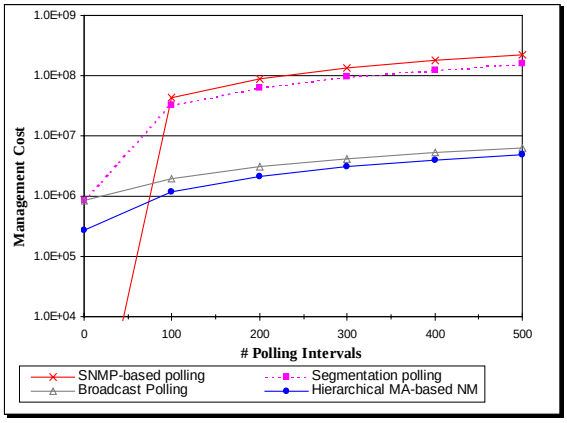


Figure 8. Comparison of the management costs for hierarchical, SNMP-based, "segmentation" and "broadcast" polling

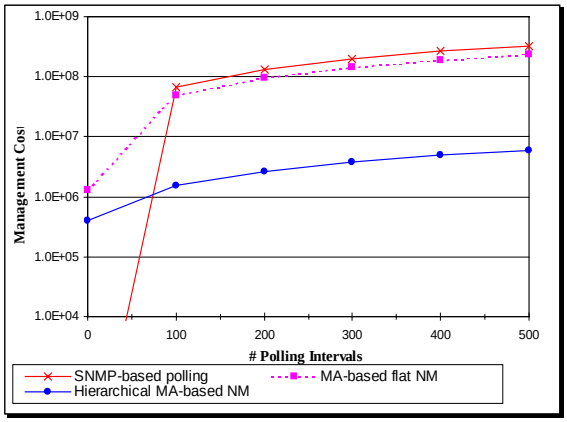


Figure 9. Comparison of the management costs when the cost coefficients are proportional to the inverse of link bandwidth

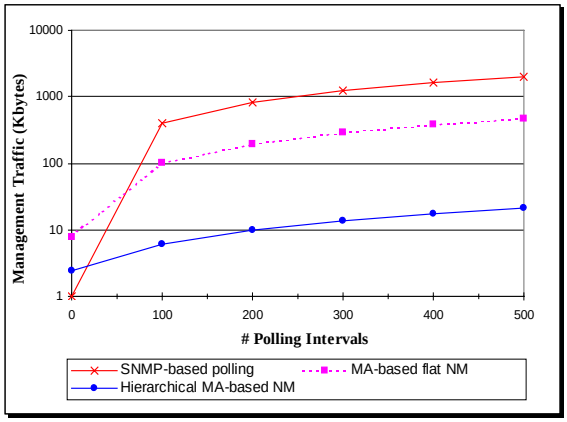


Figure 10. Bandwidth usage of the WAN link imposed by hierarchical, SNMP-based and flat MA-based management

As shown in Figure 8, the introduction of "segmentation" polling (see Figure 1c) slightly improves the performance of flat NM in terms of the management cost, while it greatly reduces the overall response time, as evidenced in [8]. We have also measured the management cost for "broadcast" polling (see Figure 1d), where the MAs are assumed to remain on the managed devices for *10* PIs, before returning to the manager host to deliver an equal number of samples. Although the cost is reduced compared to "segmentation" scheme, it still does not fall below that of hierarchical management. In addition, "broadcast" polling cannot be utilised in time-critical applications.

Figure 9 illustrates the same comparison as Figure 7, with the difference that the cost coefficients are now proportional to the inverse of link bandwidths. Assuming that all the subnetworks are 10 Mbps Ethernet and that the bandwidths of the links connecting Subnet 0 to Subnet 1 and Subnet 2 are 10 Mbps and 64 Kbps respectively, the cost coefficients become: $k_{S_0,S_0} = k_{S_1,S_1} = k_{S_2,S_2} = k_{S_0,S_1} = 1$ and $k_{S_0,S_2} = 156.25$ (when multiplied with a normalising factor $c_{nor} = 10^7$). The increase of the WAN link coefficient value amplifies the separating gap between the cost of hierarchical NM and these of SNMP-based and flat management, although this is not visible due to the logarithmic scale.

Figure 10 focuses on the NM traffic generated from each of the compared paradigms on the WAN link connecting Subnet 0 to Subnet *2*. Again, the hierarchical NM framework outperforms both flat MA-based and SNMP management with sufficient distinct. In

particular, following bytecode distribution and MDM deployment, our framework uses the WAN link only to deliver the statistics to the manager host, every *10* PIs. In contrast, SNMP heavily utilises the link to broadcast request messages and receive back the associated responses, while in flat management an MA object traverses the link at least twice in every PI, provided that MAs itineraries are optimised so as to poll the remote LAN hosts in sequence.

Although our prototype has not yet been tested in a large enterprise network environment, it is expected to scale well, especially when combining our hierarchical framework with "segmentation"/ "broadcast" polling schemes. For instance, MDMs could be easily coupled with "segmentation" scheme: after deploying the MDMs to their remote domains (possibly comprising large numbers of hosts), these could launch sufficient MAs per PI in order to reduce the overall response time and thereby further improve system scalability.

It is also worth noting that our testbed uses quite powerful nodes (PCs) and has not been tested on real mobile hardware (PDAs, mobile phones, laptops) with limited processing power and/or memory. The requirements of our platform on CPU/memory resources of managed devices are determined by the cost of running an SNMP agent (fairly lightweight), a Java Virtual Machine (JVM), a MAS and occasionally some MAs (acting either as data collectors or as MDMs). The MASs and the MAs have very low requirements on computational resources. Furthermore, the cost of running a JVM on a device is decreasing as Java chips technology is evolving in the direction of ever-smaller footprints (e.g., picoJava microprocessors can be used to run applications in small electronic appliances such as organisers, pagers, and cell phones [18]).

9. Conclusions

This paper introduces the concept of *adaptive* hierarchical management and provides a rationale for the use of MA technology. The proposed hierarchical architecture is intrinsically dynamic by employing mobile mid-level managers (MDMs) that may transparently move to a specific network domain to take over its management responsibility and localise the associated traffic. Although hierarchical MA-based management is not an entirely new concept (see [14][21][26]), our infrastructure goes one step beyond by offering improved adaptability to changing networking environments and defining concrete policies regarding network segmentation into management domains, MDMs deployment and explicit determination of domain boundaries.

Apart form their ability to move from a management domain to another, MDMs may also move within their managed domain. In particular, MDMs periodically inspect the resource availability of their managed nodes and choose to move and resume execution to the least loaded host, allowing for more balanced distribution of processing and memory load. In addition to addressing flexibility issues, management scalability is also further improved by fully exploiting the benefits of agent mobility. MDMs rely on other MAs for the data collection process; these MAs apply filtering operations locally, thereby minimising the volume of data transferred within the individual management domains.

Despite its advantages, our approach is likely to involve a high number of MAs (MDMs) which rises MAs management issues. In addition , the fault-tolerance features of our platform should be enhanced so as to cope with failures on the nodes where MDMs execute.

It is emphasised that our proposed design ideas are supplemented by a complete prototype implementation. An analytical quantitative evaluation, oriented to our specific framework design, has been described, in which functions defining the management cost associated with the proposed architecture against that of centralised NM have been derived. A prototype implementation of the introduced MAF has been tested in a realistic topology scenario, comparing its performance against both centralised and flat MA-based NM. The results section, which builds upon and complements the quantitative evaluation study, along with measurements extracted from our experimental test-bed, shows that the proposed architecture outperforms other candidate approaches with sufficient distinction, both in terms of the overall management cost and the bandwidth usage of low-bandwidth WAN links.

List of Acronyms

CMIP: Common Management Information Protocol
CPU: Central Processing Unit
CS: Client/Server
DISMAN: Distributed Management
GnG: Get 'n' Go
GnS: Go 'n' Stay
GUI: Graphical User Interface
IETF: Internet Engineering Task Force
JVM: Java Virtual Machine
LAN: Local Area Network
M2M: Manager-to-Manager
MA: Mobile Agent
MAC: Medium Access Control
MAF: Mobile Agent Framework
MAG: Mobile Agent Generator
MAS: Mobile Agent Server
MbD: Management by Delegation
MDM: Mobile Distributed Manager
MIB: Management Information Base
MLM: Middle Level Manager
NE: Network Element
NM: Network Management
OSI: Open Systems Interconnection
PDA: Personal Digital Assistant
PDU: Protocol Data Unit
PI: Polling Interval
PT: Polling Thread
RMI: Remote Method Invocation
RMON: Remote Monitoring
SNMP: Simple Network Management Protocol
WAN: Wide Area Network

References

[1] Baldi M., Gai S., Picco G.P.. "Exploiting Code Mobility in Decentralized and Flexible Network Management", Proceedings 1st Int. Workshop on Mobile Agents (MA'97), LNCS 1219, Springer, pp. 13-26, April 1997.

[2] Baldi M., Picco G.P., "Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications", Proceedings of the 20th Int. Conf. on Software Engineering (ICSE'98), pp. 146-155, April 1998.

[3] Bellavista P., Corradi A., Stefanelli C., "An Open Secure Mobile Agent Framework for Systems Management", Journal of Network and Systems Management (JNSM), 7(3), September 1999.

[4] CORBA/IIOP 2.2 Specification, http://www.omg.org/library/corbaiiop.html.

[5] Corradi A., Stefanelli C., Tarantino F., "How to Employ Mobile Agents in Systems Management" Proceedings of the 3rd Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'98), pp. 17-26, March 1998.

[6] Farmer W., Guttman J., and Swarup V., "Security for mobile agents: Issues and requirements", Proceedings of the 19th National Information Systems Security Conference, pp. 591-597, October 1996.

[7] Gavalas D., Greenwood D., Ghanbari M., O'Mahony M., "An Infrastructure for Distributed and Dynamic Network Management based on Mobile Agent Technology", Proceedings of the IEEE Int. Conf. on Communications (ICC'99), pp. 1362-1366, June 1999.

[8] Gavalas D., Greenwood D., Ghanbari M., O'Mahony M., "Complimentary Polling Modes for Network Performance Management Employing Mobile Agents", Proceedings of the IEEE Global Communications Conference (Globecom'99), pp. 401-405, December 1999.

[9] Gavalas D., Greenwood D., Ghanbari M., O'Mahony M., "Advanced Network Monitoring Applications Based on Mobile/Intelligent Agent Technology", Computer Communications, 23(8), pp. 720-730, April 2000.

[10] Goldszmidt G., Yemini Y., Yemini S., "Network Management by Delegation", Proceedings of the 2nd Int. Symposium on Integrated Network Management (ISINM'91), April 1991.

[11] ISO/IEC 9596, Information Technology, Open Systems Interconnection, Common Management Information Protocol (CMIP) – Part 1: Specification, Geneva, Switzerland, 1991.

[12] Java Remote Method Invocation (RMI), http://java.sun.com/products/jdk/rmi/ index.html.

[13] Levi D., Schoenwaelder J., "Definitions of Managed Objects for the Delegation of Management Scripts", RFC 2592, May 1999.

[14] Liotta A., Knight G., Pavlou G., Modelling Network and System Monitoring Over the Internet with Mobile Agents", Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'98), pp. 303-312, February 1998.

[15] McCloghrie K., Rose M., "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", RFC 1213, March 1991.

[16] Nicklisch J., Quittek J., Kind A., Arao S., "INCA: An Agent-Based Network Control Architecture", Proceedings of the 2nd Int. Workshop on Intelligent Agents for Telecommunication Applications (IATA'98), LNCS vol. 1437, Springer, pp. 143-155, July 1998.

[17] Oliveira J.L., Lopes R.P., "Distributed Management Based on Mobile Agents", Proceedings of the 1st Int. Workshop on Mobile Agents For Telecommunication Applications (MATA'99), pp. 243-258, October 1999.

[18] Sun Microsystems, picoJava technology, http://www.sun.com/microelectronics/communitysource/picojava/.

[19] Pualiafito A., Tomarchio O., Vita L., “MAP: Design and Implementation of a Mobile Agents Platform”, Journal of Systems Architecture, 46(2), pp.145-162, January 2000.
[20] Rubinstein, M., Duarte O. C., Pujolle G., "Reducing the Response Time in Network Management by Using Multiple Mobile Agents", accepted to the 4th Int. Conf. on Autonomous Agents (Agents’2000), June 2000.
[21] Sahai A., Morin C., “Enabling a Mobile Network Manager through Mobile Agents”, Proceedings of the 2nd Int. Workshop on Mobile Agents (MA’98), LNCS vol. 1477, Springer, pp. 249-260, September 1998.
[22] SNMP research, “The Mid-Level Manager”, http://www.snmp.com/products/mlm.html.
[23] Stallings W., “SNMP, SNMPv2, SNMPv3 and RMON 1 and 2”, 3rd ed., Addison Wesley, 1999.
[24] Susilo G., Bieszczad A., Pagurek B., “Infrastructure for Advanced Network Management based on Mobile Code”, Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS’98), pp. 322-333, February 1998.
[25] Waldbusser S., “Remote Network Monitoring Management Information Base”, RFC 1757, February 1995.
[26] Zapf M., Herrmann K., Geihs K., “Decentralized SNMP Management with Mobile Agents”, Proceedings of the 6th IFIP/IEEE Int. Symposium on Integrated Network Management (IM’99), pp. 623-635, May 1999.