Deploying a Hierarchical Management Framework Using Mobile Agent Technology

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

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

[*]Distributed Network Management and Agent Technology Research Group
Fujitsu Telecommunications Europe Ltd.,
Northgate House, St. Peters Street, CO1 1HH, Colchester, U.K.
E-mail: D.Greenwood@ftel.co.uk

Abstract. The use of Mobile Agent (MA) paradigm has been proposed by many researchers as an answer to the scalability and flexibility limitations of centralised Network Management (NM). Nevertheless, while large enterprise networks are already hierarchically structured, MA-based management has not yet moved from 'flat' to hierarchical structures. That results in non-scalable flat architectures, particularly when the management of remote subnetworks is considered. In this context, the deployment of a hierarchically structured MA-based management framework is a reasonable approach. The migration to hierarchical structures is achieved with an additional management entity, the Mobile Distributed Manager (MDM), which takes the full control of managing a given network segment. This architecture exploits the mobility features of MDMs to dynamically adapt to mutable networking conditions. Empirical results indicate a substantial reduction of the overall management cost compared to both centralised and MA-based flat management approaches.

1. Introduction

Contemporary large enterprise networks span applications, 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. Accordingly, the design of such networks' management system needs to be aligned with the corresponding managed network hierarchical structures. Hence, several hierarchical/distributed management solutions have been proposed both by researchers [1][2] and standardisation forums [3][4].

Despite these efforts towards management distribution, the majority of traditional systems still rely on centralised architectures, wherein operational data is collected by stationary *agents* embedded in network devices and subsequently gathered by a central platform (*manager*) using a management protocol, such is the IETF Simple

Network Management Protocol (SNMP) [5]. The centralised nature of NM protocols results in massive transfers of NM data that stress the network resources to their limits, while causing processing bottlenecks at the manager host. Another serious disadvantage of centralised paradigm is its intrinsic architecture inflexibility as the functionality of both managing and managed parties is rigidly defined at design time.

Hierarchical NM models help to overcome the scalability limitations of their centralised counterparts through delegating part of the management functionality to dual-role entities; these are responsible for a set of devices, which play the role of the agent when managers request information, while acting as managers for the agents located within their domain. Yet, these models cannot sufficiently meet the flexibility requirements of today's networks, with the fluctuating traffic patterns and topology structures. This is due to the static definition of the incorporated NM components and their corresponding roles in the management hierarchy. Such static configurations would possibly provide a feasible solution for moderately small and/or not very dynamic networks. However, it is not in step with the evolution of large-scale enterprise networks.

In search of more flexible management solutions, the Mobile Agents (MA) paradigm has recently attracted considerable attention in the field of distributed NM. MAs introduce a new software communication paradigm that allows code migration between hosts for remote execution. However, there is a notable inconsistency between the currently hierarchically structured networks and the emerging Mobile Agent Frameworks (MAF) that insist on 'flat' models. These models bring about scalability issues when the management of large networks is considered, whilst resulting in heavy use of low-bandwidth WAN links connecting remote subnetworks to the backbone network, due to the frequent MA transfers.

This work attempts to address these issues by coupling the concepts of hierarchical management and Mobile Agents. Therefore, we propose the deployment of a hierarchically structured, dynamic MA-based management infrastructure. To attain this objective, we introduce a novel management entity termed the "*Mobile Distributed Manager*" (MDM), which operates at an intermediary level between the manager and the stationary agents. MDMs are essentially MAs that may be dynamically assigned to / removed from a network domain in response to a change in traffic distribution or network's topology. Upon their migration, MDMs take full control of managing the assigned domain, localising the associated management traffic. In addition, MDMs may transparently move within their domain when their hosting processor becomes overloaded, thereby optimising the usage of local resources.

The rest of the paper is organised as follows: Section 2 comprises an overview of several MAFs employed to distributed NM applications, whilst Section 3 explains the rationale and introduces our proposed hierarchical approach. Section 4 provides an overview of the MAF used to support this work. Section 5 discusses the implementation details of the introduced architecture with a performance evaluation given in Section 6 and conclusions drawn in Section 7.

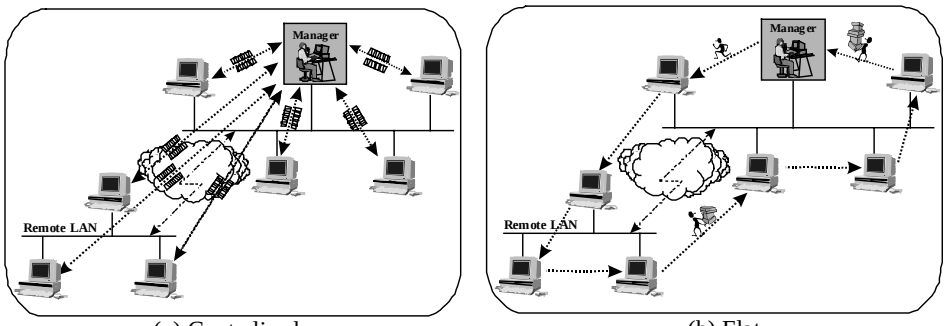
2. Mobile Agents-based Distributed Management

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 [6][17]. An exhaustive review of all the MAFs proposed for distributed NM in the last few years is beyond the scope of this paper, so we have chosen to concentrate on the most representative.

Thus, several works [6]-[12] deal with implementations of MAFs employed for NM, while others [13][14] are confined to describing the general concepts behind the introduced frameworks and discussing the arising theoretical aspects. Another set of research papers [15][16] include performance evaluations of MA-based management.

Focusing on the works supplemented by implementations, many researchers [6]-[11] have deployed new MAFs from scratch whereas some others [12] have chosen general-purpose, commercial platforms to comprise the core of their architectures. In the majority of the former [7]-[10], MAs can interact with standard SNMP agents providing access to legacy systems. Currently, only the frameworks described in [10] and [11] comply with the OMG *Mobile Agent System Interoperability Facility* (MASIF) [18] emerging standard that defines agent mobility and management policies. In addition, Java programming language [19] has dominated, being the implementation platform in all cases.

Regarding their practical use within NM, MAs have been utilised to address a broad spectrum of applications, among others: network monitoring [11][13]; traffic analysis [12]; collecting atomic Management Information Base (MIB) object values from multiple hosts [8]; calculating aggregation functions combining several MIB values [9][10]; obtaining snapshots and filtering the contents of SNMP tables [17]; automated fault management of SDH networks [6], etc.



(a) Centralised (b) Flat

Fig. 1. Centralised vs. 'flat', MA-based Network Management

The common denominator below the aforementioned MA-based management architectures is the assumption of a 'flat' network architecture, i.e. a single MA is

launched from the manager platform and sequentially visits all the managed NEs, regardless from the underlying topology (see Figure 1b). 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, for large networks the round-trip delay for the MA will greatly increase, whilst the network overhead may overcome that of centralised paradigm (the MA size will grow after visiting each of the nodes included into its itinerary [15]). The situation seriously deteriorates when considering management of remote LANs, connected to the main site through low-bandwidth, expensive WAN links. In this case, frequent MA transfers may potentially create bottlenecks and considerably increase the management cost. Hence, the deployment of a hierarchically structured MA-based management framework seems a rational approach to overcome this problem.

3. Mobile Agent-based Approach to Hierarchical Management

Surprisingly, MA-based architectures proposed for distributed NM have not yet moved from flat to hierarchical structures. This represents an inconsistency though, as these management systems are not directly mapped to the managed networks. In this work we address this issue through combining the concepts of Hierarchical/Distributed Management and Mobile Agents for NM. Specifically, we introduce the 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 (see Figure 2) and takes over the management of local NEs from the central manager. In a later section, we discuss how the decision concerning the selection of the host, where the MDM will carry out its management tasks from, is made.

As a result, the traffic related to the management of that domain will be localised, as the MDM will be able to dispatch and receive MAs to collect NM data from the local hosts, or even execute centralised management operations on them. The MDM will continue to perform its tasks without the manager's intervention, even if the interconnecting link fails. A first-line response will also be 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 mobility feature of MDMs allows the management system to adapt dynamically to a mutable environment, optimising the use of network resources. Apart from the fact that management functionality may be added/configured at runtime, this architecture can also dynamically adapt to changing networking conditions. Namely, an MDM entity can be deployed to / removed from a network segment to reflect a change on traffic patterns, or move to the least loaded host to minimise the usage of local resources.

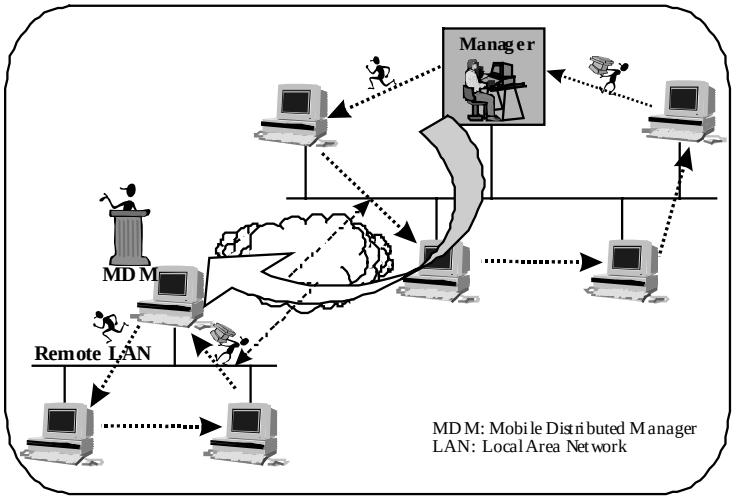


Fig. 2. Hierarchical MA-based management

Similar work has been reported in two research papers. Liotta et al. [13] have conducted an interesting study of an MA-based management architecture adopting a hierarchical, multi-level approach. Interesting cost functions corresponding to various MA configurations are also discussed. However, there is no implementation supplementing this work, while the authors have not considered providing "*Middle Managers*" mobility features, so as to dynamically change their location. In addition, the criteria according to which the managed network is segmented in domains and the way that these domains are assigned to Middle Managers are not explicitly mentioned. In [14], Oliveira and Lopes proposed the integration of the IETF's *Disman* framework [4] into their MA-based NM infrastructure. Again, this work lacks implementation details while the "*Mobile Disman*" architecture they propose is heavyweight (it consists of many resource-demanding components) and would certainly have increased requirements on system resources.

4. Overview of 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 four major components [9], 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 relating to the NEs. Active agent processes 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 resides logically above the standard SNMP agent, creating an

efficient run-time environment for receiving, instantiating, executing, and dispatching MA objects. Integration with SNMP was vitally important to maintain compliance with the legacy management systems.

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.

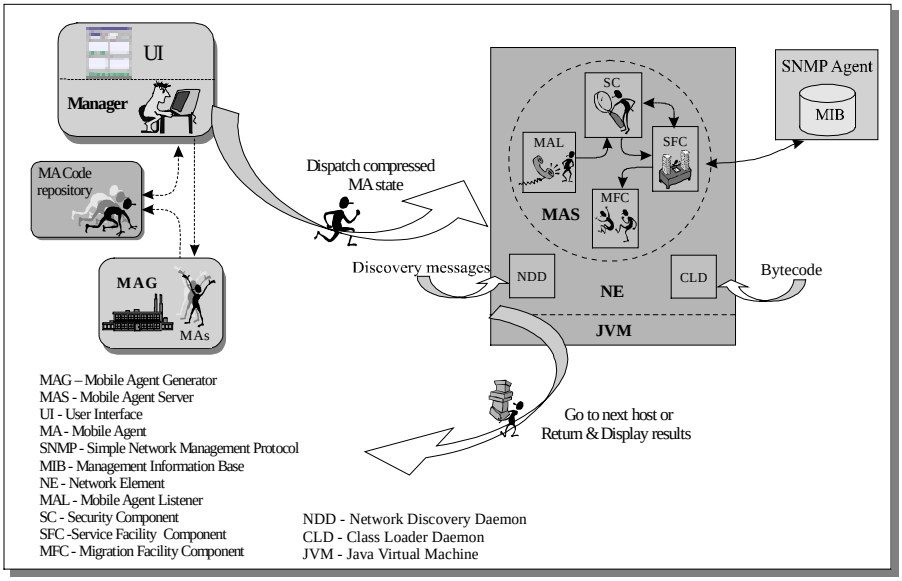


Fig. 3. The Mobile Agents-based NM Infrastructure

(III) Mobile Agent Generator (MAG): The MAG is essentially a tool for automatic MA code generation allowing the construction of customised MAs in response to service requirements. Generated MA code is stored into an MA code repository (see Figure 3). Such MAs may dynamically extend the infrastructure's functionality, post MAS initialisation, to accomplish management tasks tailored to the needs of a changing network environment.

The MAG's operation is described in detail in [9]. However, its functionality has been extended so as to allow the operator (through a dedicated GUI) to specify: the polling frequency (i.e. the polling interval's duration); the transmission protocol to be used for the MA transfers (either TCP or UDP); the security policies.

(IV) 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 [17], 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.

5. Implementation Details

5.1. Topology Map

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 presents 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 "*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 Fig. 4.). Each of the tree nodes consist of the following attributes:

- subnetwork's name;
- the names of hosts and routers 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 elements, each corresponding to a specific host, that will be made visible upon discovering an active MAS entity on that host.

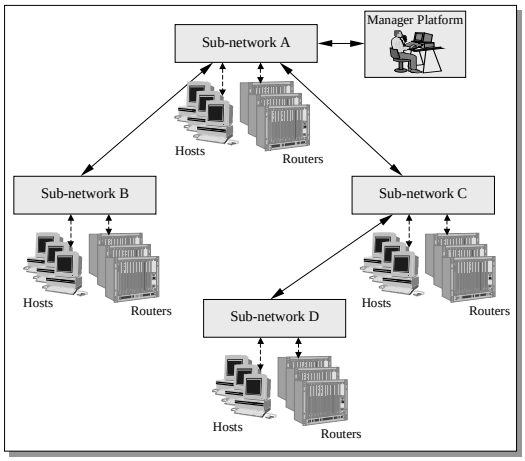


Fig. 4. The topology tree structure

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

$$n_{s,subA} = n_{l,subA} + n_{l,subB} + n_{l,subC} + n_{l,subD} \quad (1)$$

As described in a following section, 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.

5.2. Policies on MDMs Deployment

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.

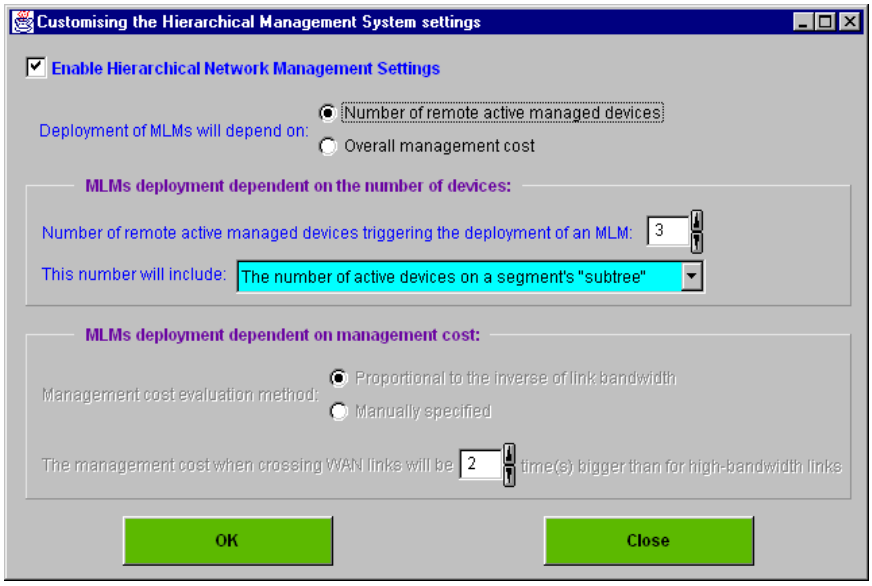


Fig. 5. Graphical User Interface for customising the hierarchical NM system policies

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. 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 management cost.

In the former case (*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 $n_{l,S} \geq N$, otherwise to every segment with $n_{s,S} \geq N$. In the latter case (*Policy 2*), the management cost may either be: (a) proportional to the inverse of link bandwidth, or (b) manually specified.

5.3. Implementing MDMs Deployment

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. A similar procedure is followed when a MAS server is being shut down.

The discovery or termination of a MAS server triggers an event at the manager host. The topology tree is then scanned with the subnetworks that meet specific 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) 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. Upon its arrival at the remote subnetwork, the MDM instantiates the PTs, which thereafter start performing their tasks without any further disruption of the management process.

5.4. MDM's Migration Within its Domain

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.

This is accomplished through the regular inspection of the other 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. If the hosting processor is seriously overloaded, compared to the neighboring devices, the MDM will transparently move to the least loaded node. In our early prototype, MAs may only extract memory usage values, but will soon be able to acquire CPU load information also.

5.5. Communication Between Manager and MDMs

One of the key advantages of our framework 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 require an MDM to terminate its execution or move to another domain, to download in runtime an additional management service, i.e. a new MA object along with its corresponding PT configuration, etc.

We have chosen Java RMI [20] 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. Performance Evaluation

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 [13]. 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 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 polling intervals (PI) would be:

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

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:

$$C_{hier} = C_{distr} + C_{depl} + C_{pol} + C_{deliv} \qquad (3)$$

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.

Concerning bytecode distribution, we adopt a lightweight scheme: in contrast with all known MAFs proposed for management applications [6]-[11], wherein both the MA's code and state are transferred on each migration, we have chosen to distribute the bytecode at the MA's construction time and thereafter transfer only the state information, resulting in a much lower demand on network resources (bytecode size is typically much larger than state size [15]). The code distribution scheme proposed in [6]-[11] 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 polling intervals.

The introduction of MDMs reduces the code distribution cost even further: the MAs bytecode is no longer broadcasted to all managed devices, as in flat management [9], 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:

$$C_{distr} = \left(k_{S_0,S_0} * N_0 + \sum_{i=0}^{M} \left[k_{S_0,S^i} + k_{S^i,S^i} * N_i \right] \right) * C \qquad (4)$$

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:

$$C_{depl} = \sum_{i=0}^{M} k_{S_0,S^i} * ST_0 \qquad (5)$$

where ST_i represents the compressed state size of an MA when migrating from the i^{th} host.

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:

$$C_{pol} = \left(\sum_{i=0}^{m} k_{S^i,S^{i+1}} * ST_i + \sum_{i=0}^{M} \sum_{j=0}^{N_i} k_{S^i,S^i} * ST_j \right) * p \qquad (6)$$

Clearly, the first term of the summation dominates 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^{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 [13] 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:

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

The last term appearing in Eq. (6-2) represents the cost associated with the delivery of the gathered data from the MDMs to the manager host:

$$C_{deliv} = \left(\sum_{i=0}^{M} k_{S^i, S_0} * D/t \right) * p \tag{8}$$

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

The quantitative model introduced in this section has been applied to the test network shown in Figure 6, where the network domain margins are depicted by the dotted curved lines.

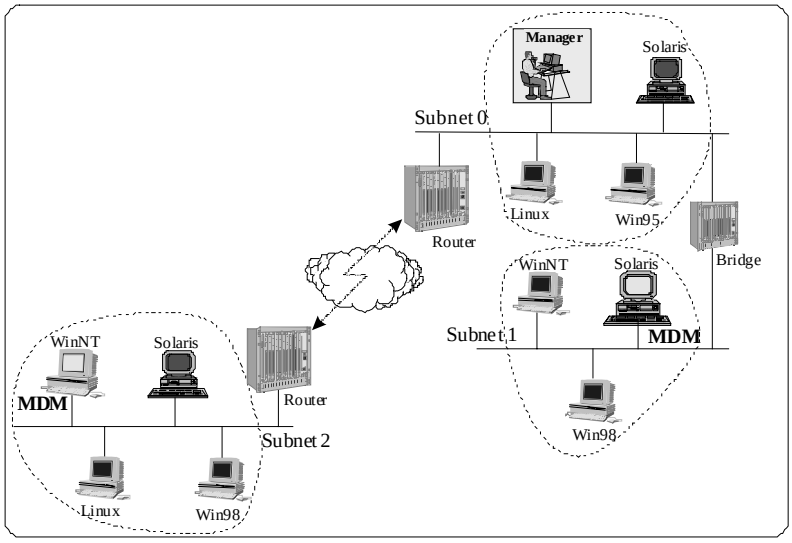


Fig. 6. The test network

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 $k_{S_0,S_2} = 50$. These values are chosen in accordance to the bandwidth of the links they correspond to. We have also measured the set variables: S_{req} = 90 bytes, ΔS_{req} = 17 bytes, *C* = 1.95 Kb, and ST_0 =

447 bytes. These values have been measured after testing our framework in a real network comprising Solaris and WinNT devices.

Equations (6-1)-(6-7) 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 7a, drawn on a logarithmic scale. The functions defining the cost of MA-based flat management represent special cases of those developed for hierarchical management.

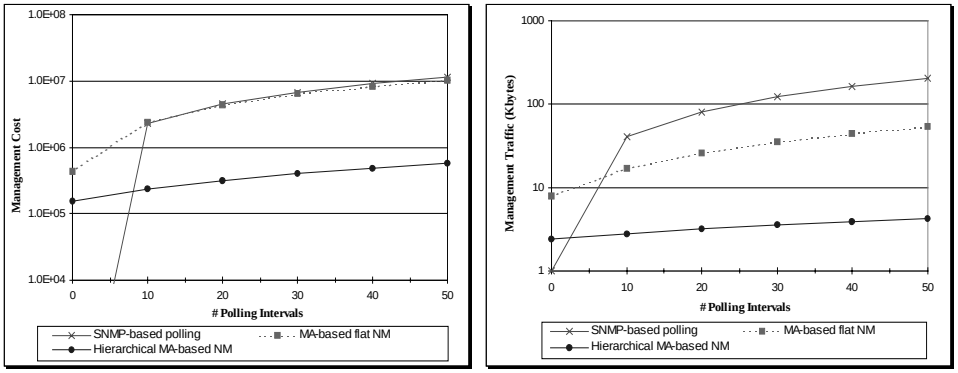


Fig. 7. Comparison between SNMP-based polling, flat MA-based polling and the proposed hierarchical framework in terms of: (a) Overall management cost, (b) Bandwidth usage of the WAN link

We consider a data intensive application, namely polling every host for the contents of the MIB-II *interfaces* table [21]. We assume the minimum of *two* interfaces per host, i.e. 2×21 collected values per host, since each table row includes *21* columns. As described in [17], the MA objects are capable 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. That results in improving system scalability, due to the low selectivity ratio $\sigma \approx 1.82\%$ achieved over the obtained data (b = 714 bytes/host). In other words, the MAs state size increases only by $\sigma \times b = 13$ bytes, for each visited host. We also assume that management data (D = 39 bytes/PI for Subnet 1 and 52 bytes/PI for Subnet 2) are delivered to the manager from distributed MDMs with a frequency of t = 10 PIs.

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 above.

Figure 7b 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.

7. Conclusions – Future Work

This paper has proposed the use of MA technology for dynamic hierarchical management. In this context, we introduced the MDM, a novel management entity, which is dynamically assigned to a given network segment and localises the associated management traffic. MDMs mobility feature allows the management system to adapt to potential changes of the managed network topology or traffic distribution and optimise the use of local resources. Finally, a performance evaluation in terms of the overall management cost confirms the proposed model's improved scalability over both traditional centralised and flat MA-based architectures.

Future work will address the following issues:

- Optimisation of MAs itinerary, so that in case the manager or an MDM manages more than one subnetwork, the MAs will not traverse the interconnecting links more than twice.
- Investigate the use of CORBA [22] instead of RMI to implement the communication bus between distributed MDMs and the manager host.

List of Acronyms

CORBA: Common Object Request Broker Architecture
CS: Client/Server
DM: Distributed Manager
GUI: Graphical User Interface
IETF: Internet Engineering Task Force
LAN: Local Area Network
MA: Mobile Agent
MAC: Medium Access Control
MAF: Mobile Agent Framework
MAG: Mobile Agent Generator
MAS: Mobile Agent Server
MASIF: Mobile Agent System Interoperability Facility
MbD: Management by Delegation
MDM: Mobile Distributed Manager
MIB: Management Information Base
NE: Network Element
NM: Network Management
OSI: Open Systems Interconnection
PI: Polling Interval
PT: Polling Thread
RMI: Remote Method Invocation
RMON: Remote Monitoring
SNMP: Simple Network Management Protocol
WAN: Wide Area Network

References

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

[2] Siegl M. R., and Trausmuth G., "Hierarchical Network Management: A Concept and its Prototype in SNMPv2", Computer Networks and ISDN Systems, Vol. 28, No. 4, pp. 441-452, February 1996.
[3] S. Waldbusser, "Remote Network Monitoring Management Information Base", RFC 1757, February 1995.
[4] Distributed Management (disman) Charter, http://www.ietf.org/html.charters/disman-charter.html.
[5] Case J., Fedor M., Schoffstall M., Davin J., "A Simple Network Management Protocol (SNMP)", RFC 1157, May 1990.
[6] Sugauchi K., Miyazaki S., Covaci S., Zhang T., "Efficiency Evaluation of a Mobile Agent Based Network Management System", 6th International Conference on Intelligence and Services in Networks (IS&N'99), LNCS vol. 1597, pp. 527-535, April 1999.
[7] 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.
[8] Sahai A., Morin C., "Enabling a Mobile Network Manager through Mobile Agents", Proceedings of the In Proceedings of the 2nd International Workshop on Mobile Agents (MA'98), LNCS vol. 1477, pp. 249-260, September 1998.
[9] 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 International Conference on Communications (ICC'99), pp. 1362-1366, June 1999.
[10] Pualiafito A., Tomarchio O., Vita L., "MAP: Design and Implementation of a Mobile Agents Platform", to appear in Journal of System Architecture.
[11] Bellavista P., Corradi A., Stefanelli C., "An Open Secure Mobile Agent Framework for Systems Management", Journal of Network and Systems Management (JNSM), Special issue on Mobile Agent-based Network and System Management, Vol. 7, No 3, September 1999.
[12] Feridun M., Kasteleijn W., Krause J., "Distributed Management with Mobile Components", Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management (IM'99), pp. 857-870, May 1999.
[13] 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.
[14] Oliveira J.L., Lopes R.P., "Distributed Management Based on Mobile Agents", Proceedings of the 1st International Workshop on Mobile Agents For Telecommunication Applications (MATA'99), October 1999.
[15] Fuggetta A., Picco G.P., Vigna G., "Understanding Code Mobility", IEEE Transactions on Software Engineering, Vol. 24, No. 5, pp. 342-361, 1998.
[16] Rubinstein M., Duarte O.C., "Evaluating Tradeoffs of Mobile Agents in Network Management", Networking and Information Systems Journal, Vol. 2, No. 2, July 1999.
[17] Gavalas D., Greenwood D., Ghanbari M., O'Mahony M., "Advanced Network Monitoring Applications Based on Mobile/Intelligent Agent Technology", to appear in Computer Communications Journal.
[18] GMD Fokus, IBM Corp., "The OMG MASIF Standard", http://www.fokus.gmd.de/research/cc/ima/masif/.
[19] Sun Microsystems: "Java Language Overview – White Paper" [On-line] (1999), URL: http://www.javasoft.com/docs/white/index.html.
[20] Java Remote Method Invocation (RMI), http://java.sun.com/products/jdk/rmi/index.html.
[21] McCloghrie K., Rose M., "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", RFC 1213, March 1991.
[22] CORBA/IIOP 2.2 Specification, http://www.omg.org/library/corbaiiop.html.