Mobile Software Agents for Decentralised Network & Systems Management

Damianos Gavalas, Dominic Greenwood, Mohammed Ghanbari, Mike O'Mahony

Abstract--**Mobile Agent (MA) technology has been proposed for Network & Systems Management (N&SM) as an answer to the scalability limitations of centralised models and the flexibility problems of static hierarchical frameworks. Yet, much still need to be done to deploy MA-based management frameworks that efficiently cope with the dynamically changing traffic and topological characteristics of modern networks. This paper starts with an extensive review of the research approaches on MA-based management. Then, we propose a highly adaptive and dynamic hierarchical MA-based framework for N&SM that is based on appropriate policies to enable automatic calibration of the management system depending on network conditions. The applicability of the introduced framework is tested in realistic management scenarios and three applications on network monitoring are proposed.**

***Index Terms*--Distributed Network Management, Mobile Agents, Network Monitoring.**

I. Introduction

Mobile Agents (MA), defined as autonomous programs with the ability of moving from host to host and acting on behalf of users towards the completion of a given task, attract increasing attention within the distributed computing field. One of the most popular topics in MA research community has been distributed Network & Systems Management (N&SM).

Traditionally, N&SM systems rely on centralised, Client/Server approaches wherein the functionality of both clients (managers) and distributed servers (agents) is defined at design time. This centralised model is exemplified in the IETF Simple Network Management Protocol (SNMP) [16]. Within SNMP, physical resources are represented by managed objects with the latter being grouped into standard or proprietary tree-structured Management Information Bases (MIB). The centralised paradigm is known to exhibit severe scalability problems as it involves massive transfers of management data, which cause considerable strain on network throughput and processing bottlenecks at the manager host. Moreover, the system is highly dependent on the central management station. If the latter goes offline or a key network link fails, the system is no longer functional.

These problems have motivated a trend towards distributed management intelligence that represents a rational approach to overcome the limitations of centralised N&SM. In the 90's, the scene has been dominated by the Management by Delegation (MbD) [7] initiative, with decentralisation and automation of management tasks realised by dynamically delegating management functions to stationary agents ("elastic processes"), which collect and process management locally. Along the same line, several hierarchical systems have been proposed that involve the introduction of "mid-level manager" [14] entities. Such entities are responsible for few agents, collecting row data from them, performing some calculations and producing more meaningful values that can be used by a superior manager. This method significantly reduces the volume of NM traffic since only high-level information is sent to the manager.

In the standardisation arena, the trend towards management distribution was first indicated by RMON (Remote MONitoring) [16] that facilitates the collection of traffic-oriented statistics by monitoring devices (probes), which provide detailed information concerning traffic activity within their local domain. Recently, the DISMAN (DIStributed MANagement) Working Group of the IETF proposed mechanisms to delegate control above several distributed stations by distributing scripts which perform arbitrary management tasks to remote devices that implement the Script MIB [8].

However, the aforementioned approaches exhibit several limitations themselves. For instance, devices with installed MbD agents or implementing the Script MIB can only offer a limited local view of computing resources. Should information describing the state of a set of devices is required, data from several individual hosts need to be collected and correlated either from the manager station itself or from a supervising mid-level manager. In addition, RMON represents a rather inflexible approach, as the control operations of a probe may be set/modified only at configuration time. Regarding hierarchical frameworks, although they point to the right direction, they are characterised by inflexible and static definition. In particular, they imply a rigidly defined management configuration that cannot easily be modified and, therefore, is not in step with the dynamically evolving topological and traffic characteristics of large-scale enterprise networks.

MA technology can serve as a good candidate to solve

D. Gavalas is with Electronic Systems Engineering Department, University of Essex, Colchester, CO4 3SQ, U.K. (telephone: +44 1206 873498, e-mail: dgaval@essex.ac.uk).

the flexibility problems mentioned above, whilst providing scalable solutions. As a result, a number of MA Frameworks (MAF) have been proposed for N&SM [1][9][11][12][13][17][18]. MAs can be thought of as a 'superset' of MbD agents as they can offer all the functionality offered by static delegation agents, having the additional benefit of mobility. Compared to the 'static' approaches presented above, MAs offer the following advantages:

(i) Ease in modifying existing management functions as management functions are developed/modified centrally, with the modifications taking instant effect. In case that static agents are used, each of them has to be updated by an 'update' message sent from the manager to the managed devices. Frequent modifications would create a considerable amount of traffic.

(ii) Efficient use of computing resources on the managed entities, as management functions are executed only as long as the MA resides and is active on the Network Elements (NE).

(iii) Network domains boundaries are no longer pre-assigned and management components have not fixed roles and locations; MAs may enable automatic calibration of the management system upon sensing changes on network topology or traffic patterns. For instance, an MA acting as a mid-level manager could move in runtime to another host or even domain to optimise the usage of network or computing resources.

(iv) Apart from being used for simple filtering operations, the mobility of MA objects intrinsically implies a domain or global level view of the managed network as MAs visit several or even all the network hosts. That allows the MAs to apply a second stage of management data filtering, at domain or network level, i.e. merge/correlate the results acquired from each host with those already collected during their itinerary and keep only the values satisfying certain conditions. That leads to a further reduction of data transfers, while relaxing the manager host from considerable processing burden.

The remainder of the paper is organised as follows: Research approaches in MA-based distributed management are reviewed in Section 2. Section 3 describes our chosen MA-based approach to N&SM. Section 4 discusses a number of applications of our MAF in network monitoring and conclusions are drawn in Section 8.

II. Mobile Agent-Based Research Approaches on Distributed Network & Systems Management

The use of MAs 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 management archetypes. On the other hand, their use imposes extra strain on the physical resources of remote hosts, brings about performance concerns and introduces potential security threats [3].

Several MA-based frameworks proposed for network monitoring applications [11][12][13][17], 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 1a). One of the first prototypes proposed for NM has been presented in [17], where issues related with access provisioning to managed resources and communication between MAs are discussed. Pualiafito et al. [12] introduce the Mobile Agent Platform (MAP), used for monitoring the systems state by calculating aggregation functions combining several MIB values (*health* functions). MAP also complies with the OMG *Mobile Agent System Interoperability Facility* (MASIF) emerging standard [6] that defines agent mobility and management policies. An "aggregation network" of MAs is reported in [11] where MAs obtain computed views of MIB variables, delivered to certain "subscribers"; multiple levels of aggregation may be defined. Sahai & Morin [13] 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.

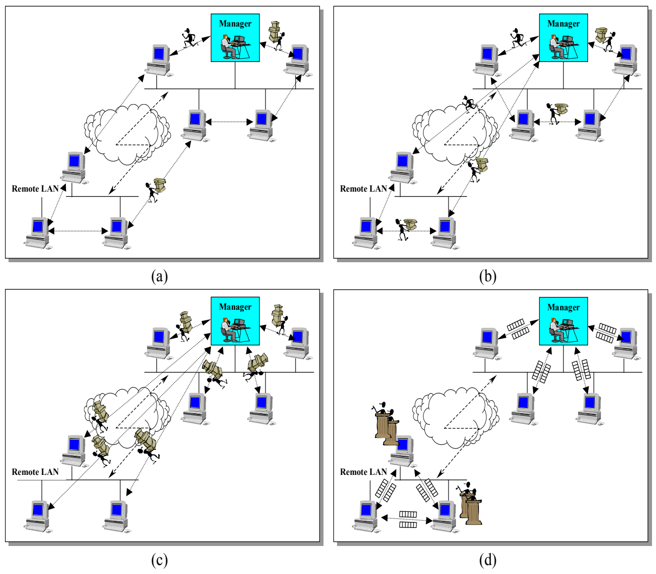


Figure 1. Centralised and distributed (MA-based) approaches to Network Management.

However, although relaxing the network from a flood of request/response SNMP messages, 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). 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 [2][5], whereby the network is partitioned into several domains and a single MA object is assigned to each of them (see Figure 1b). The MAMAS architecture introduced in [2] has been extended so as to apply a strict security scheme and address interoperability issues through compliance with CORBA and MASIF standards [1]. The "segmentation" approach introduces a high degree of parallelism in the data collection process, thereby reducing the overall response time.

Alternatively, when acquired data are to be analysed off-line, the "*broadcast*" approach [5] 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 1c). 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 1a and 1b 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 [5].

The scalability problem is more adequately addressed by hierarchical models [9][18] that have recently started coming into the picture. Liotta et al. [9] have conducted an interesting study of an MA-based management architecture adopting a multi-level approach, where MAs responsible for simple monitoring tasks may use their cloning capability for minimising deployment cost; interesting cost functions corresponding to various MA configurations are also proposed. However, [9] is not supplemented by prototype implementation. Likewise, the framework presented in [18] 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 in both [9][18], 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 1d). 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.

Furthermore, even approaches where MAs are organised in hierarchical fashion, 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 [18]. 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 at the lower hierarchy level, (between mid-level managers and the managed devices, as in [18]). Appropriate methods for processing/filtering network monitoring data at the source need therefore to be devised, especially when transfers of structures such as SNMP tables that contain large volumes of data are concerned. These methods should exploit the unique capability of MAs to move from host to host carrying with them their collected data.

Hence, the deployment of a highly adaptive hierarchically structured management model that relies on MAs both for acting as mid-level managers and collecting results seems a rational approach to address these issues and overcome the limitations of statically configured NM frameworks presented in the preceding section.

III. Hierarchical, Mobile Agent-based Network Management

In order to achieve the transition to a hierarchical model, we introduce a novel entity termed the Mobile Distributed Manager (MDM), 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 residing in that domain (Figure 2a) and takes over the management of local NEs from the central manager.

As a result, the traffic related to the management of that domain is 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 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.

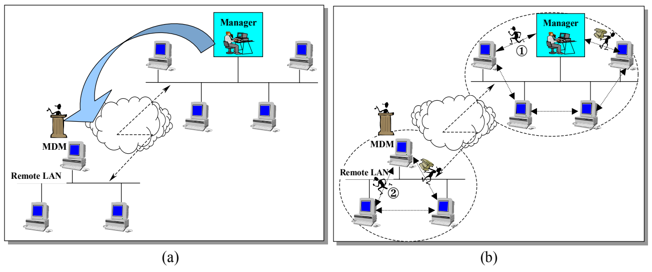


Figure 2. Hierarchical MA-based management.

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

- *Integration with existing management standards*: Our architecture provides support for the dominant Internet management protocol (SNMP), allowing interoperability with a large number of legacy management systems.
- *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 agent servers 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.
- *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*: A management system should be flexible enough to adapt to rapidly changing topological and traffic characteristics of modern networks. Hence, the location where MDMs execute is not fixed, neither is the set of hosts under their control. MDMs can transparently migrate 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, 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.

The MAF on top of which our hierarchical infrastructure has been developed, is built in Java and offers core management and mobility functionality.

A. *The Core Mobile Agent Framework*

The two main components of the core framework are the Manager application and the Mobile Agent Server (MAS) [4]. The former is 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 and maintained in a dynamically updated a 'discovered list'. MAS entities are installed on every managed device serving as an interface between visiting MAs and legacy management systems. The MASs functionally reside above standard SNMP agents, creating an efficient run-time environment for receiving, instantiating, executing, and dispatching MA objects. Security mechanisms are also integrated within the MAS modules providing *authorisation* of the MAs requests, *authentication* of incoming MAs and *encryption* of the obtained sensitive NM data.

B. *Implementation Details*

A key characteristic of our architecture is its dynamic adaptation 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 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*: (i) The population of remotely active managed devices; (ii) The overall cost involved with the management of a remote set of devices.

When applying *Policy 1*, the administrator specifies the number N of remote managed NEs that will justify the deployment of an MDM to a particular network segment. N may either denote the number of local active hosts on a subnet or in a set of subnets located in hierarchically lower levels. When the population of NEs directly managed by an MDM exceeds a certain limit, that domain will be divided to two independent domains, with a clone of the original 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.

Upon making the decision for deploying an MDM to a remote subnet, this action takes instant effect; a bundle of *TaskDescription* objects is packaged and sent along with the migrating MDM, describing the monitoring tasks to be applied from the remote location. For each one of these objects a Polling Thread (PT) is created, being responsible for a single task. The PTs will thereafter start performing their tasks (creating the required number of MAs to accomplish the task) without any further disruption of the management process.

MDMs may also autonomously move to another host, when their current hosting device is over-loaded, 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. The MDM notifies the manager application about its new location of execution before the actual migration occurs. We have chosen Java RMI 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.

IV. INTELLIGENT FILTERING APPLICATIONS OF NM DATA

The management operations defined in the IETF approach are usually very low-level, as the management station can typically only get and set atomic MIB object values. Semantically rich operations, such as *get-column*, *get-row* or *get-table* are not available yet. Using the MA-based approach, sequences of primitive operations can be grouped into higher-level operations, sent to the NEs and executed independently of the management station. This improves considerably the system's performance by reducing the number of messages exchanged between the agent and the management station, thereby limiting the network load. In the following sections, we describe three novel applications of MAs on network monitoring, demonstrating their ability to minimise management data overhead.

A. Evaluation of Health Functions

Cases often occur in polling operations, where one or two MIB variables are not a representative indicator of system state and hence an aggregation of multiple variables is required, known as a *health function* (HF). For instance, *five* MIB-II [10] objects are combined to define the percentage *E(t)* of IP output packets discarded over the total number of packets sent within a specific time interval,

$$E(t) = \frac{(ipOutDiscards + ipOutNoRoutes + ipFragFails) * 100}{ipOutRequests + ipForwDatagrams} \quad (1)$$

where MIB-II is an example of a MIB being supported by all the SNMP-enabled NEs.

In the SNMP model, the least 'expensive' option would be to group the five Object Identifiers (OIDs) in a single *get* request packet. The response packet would then include the OIDs along with the requested values, with the OIDs typically occupying more space than the actual values. On the other hand, MAs are able to compute HFs locally thereby providing a way to semantically compress large amounts of data as a single value is returned to the manager station, relieving it from processing NM data, while the MAs state size remains as small as possible. MAs can also be instructed to transmit computed values only in the case that certain thresholds are crossed.

B. Polling SNMP Tables

Some of the major drawbacks with SNMP are related to the bulk transfer of data, e.g. the transfer of large SNMP tables. When using the SNMPv1 *get-next* operator, the table retrieval requires at least one get-next operation per table row (see Figure 3a). Apart from the apparent impact on network resources, this operation is known to experience significant latency, especially when the management of remote LANs is considered [15]. In addition, the non-negligible time intervals between the acquisition of each individual object value leads to potential inconsistencies (different sections of the table will reflect updates at different times).

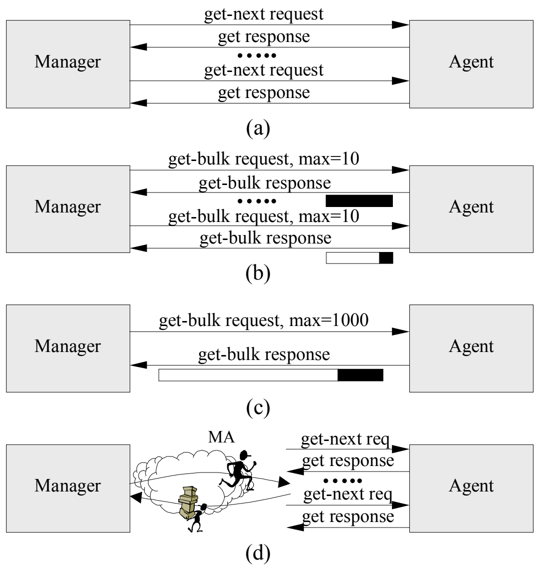


Figure 3. Acquiring an SNMP table snapshot through: (a) successive remote get-next requests, (b) multiple get-bulk requests, (c) a single get-bulk request, (d) MA migration and locally issued get-next requests.

The situation improves with the introduction of the *get-bulk* request [16], provided by later protocol versions (v2c & v3), which adds to the ability of SNMP to retrieve large blocks of data efficiently by specifying a maximum number of successive values to be returned (*max-repetitions*) [16]. That means that the human manager has to guess a value for the max-repetitions parameter. Using small numbers for max-repetitions may result in too many message exchanges (Figure 3b). Using large numbers, however, may result in an 'overshoot' effect [15]: the agent returns data that do not belong to the table the manager is interested in (Figure 3c).

Here, we propose a way to improve the retrieval of SNMP tables both in terms of network overhead and latency. An MA object is dispatched by its corresponding polling thread and visits a pre-determined number of hosts. At each place of contact, when received by the local MAS entity, the MA acquires an SNMP table through successive get-next requests (see Figure 3d). The table contents are then encapsulated into its state before moving to the next host or returning to the manager. The MA may also obtain several snapshots of the table and deliver them all to the manager through a single transfer. The overall latency is also reduced (especially for large tables), as the round-trip delay of each request/response message exchange is significantly smaller. An inviting side effect of that is the improved consistency of the acquired values.

C. *Filtering SNMP Tables*

In most existing monitoring applications, the retrieved bulk data are usually utilised to feed a processing engine responsible for extracting high-level results. These results may be used later on to aid on capturing utilisation or error rate peaks, indicate failed devices, foresee possible congestion points, for capacity planning, etc. In fact, only a small portion of the obtained values is proved useful with the rest simply being discarded as the processing action, i.e. the filtering of management data, takes place on the manager and not on the NE side.

Therefore, we propose a third application of MAs on network monitoring exploiting their ability to download management logic in order to perform intelligent filtering of NM data on selected SNMP tables. Specifically, this type of MA is able to acquire an SNMP table and subsequently apply a pre-determined filtering pattern to it.

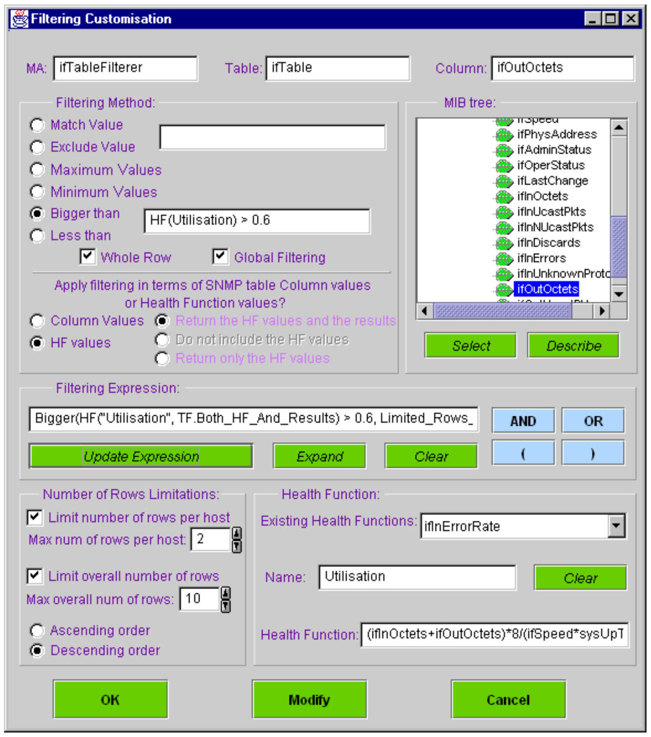
The filtering operators offered in the current implementation are classified in *Arithmetic* (Max, Min, Bigger, Less) and *Textual* (Match, Exclude). These operators typically take as input the acquired SNMP table and filter it keeping only the rows for which a given element meets certain criteria, e.g. is greater than a threshold value or matches a given text string.

When arithmetic operators are considered, the user may set limitations on the maximum number of rows that may be returned from individual hosts, the maximum overall number of rows, whether the results will be sorted in ascending or descending order, etc.

It is also noted that the filtering operation may be either based on: (i) a given table column, e.g. for the MIB-II *interfaces* table (ifTable) [10], "*get the two rows (interfaces) with the maximum number of incoming octets (max ifInOctets)*", or (ii) on a pre-defined HF, e.g. "*return the two more heavily loaded interfaces*" (see Figure 6).

For instance, the MA created by the User Interface shown in Figure 4 will return the two more heavily loaded interfaces, in descending order, given that their utilisation is greater than 60%. In particular, it will calculate the *utilisation HF* for each one of its rows:

$$U(t)=\frac{(ifInOctets+ifOutOctets)*8}{(ifSpeed*SysUpTime*100)} \qquad (2)$$


Filtering Customisation

MA: ifTableFilterer   Table: ifTable   Column: ifOutOctets

Filtering Method:
[ ] Match Value
[ ] Exclude Value
[ ] Maximum Values
[ ] Minimum Values
[✓ Bigger than] HF(Utilisation) > 0.6
[ ] Less than
[✓ Whole Row]   [✓ Global Filtering]

Apply filtering in terms of SNMP table Column values or Health Function values?
[ ] Column Values
[✓ HF values]
[✓ Return the HF values and the results]
[ ] Do not include the HF values
[ ] Return only the HF values

MIB tree: ifSpeed, ifPhysAddress, ifAdminStatus, ifOperStatus, ifLastChange, ifInOctets, ifInUcastPkts, ifInNUcastPkts, ifInDiscards, ifInErrors, ifInUnknownProto, [✓ ifOutOctets]

Filtering Expression: Bigger(HF("Utilisation", TF.Both_HF_And_Results) > 0.6, Limited_Rows_

Number of Rows Limitations:
[✓ Limit number of rows per host]
Max num of rows per host: 2
[✓ Limit overall number of rows]
Max overall num of rows: 10
[ ] Ascending order
[✓ Descending order]

Health Function:
Existing Health Functions: ifInErrorRate
Name: Utilisation
Health Function: (ifInOctets+ifOutOctets)*8/(ifSpeed*sysUpT

Figure 4. Customising the SNMP table filtering operation parameters.

The HF values are appended to the interfaces table, which is then sorted into HF values order. The resultant table is scanned and the rows (two, at maximum) with HF values greater than 0.6 are returned, in descending order. For the case where only specific table columns are desired (e.g. only the number of octets sent out of the most heavily loaded interfaces), the rest of the columns will be removed.

In addition to the simple filtering issues discussed so far, we introduce the concept of *domain* or *global level* filtering. In particular, we exploit the multi-node movement of MAs to perform an additional level (second stage) of data filtering, in domain or even in network level. This is achieved by comparing/merging the results already collected with these that have been just obtained/processed. Hence, not only is the manager host relieved from processing bottlenecks, but the MA's state size is prevented from growing rapidly (and therefore the network overhead is further reduced), since the amount of information stored in the MA's data folder basically remains constant.

It should be emphasised that global filtering may be easily coupled with both the "segmentation" and "broadcast" approaches. In the former case the MA may, for example, return the two most heavily loaded interfaces found in the entire *network*, whereas in the latter, record a utilisation peak on a host within a given observation period. When employing "broadcast" polling, higher sample rates may be used without putting any additional load on the network (since this will not affect the MA's state). In "segmentation" approach, the results will be delivered to the manager and then displayed on a graphical table component, with the interface information drawn in different colours, depending on the host they are arriving from. This filtering operation is summarised in the Figure 5 flow diagram.

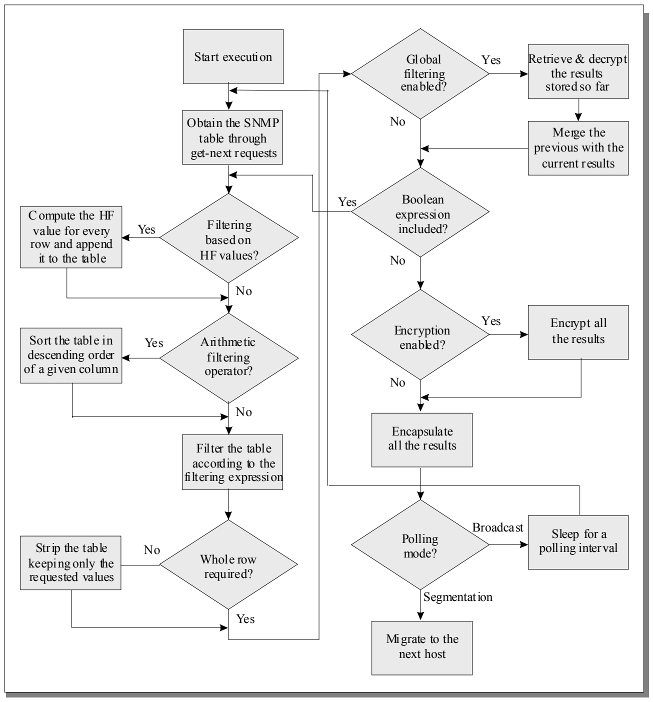

Start execution
Obtain the SNMP table through get-next requests
Filtering based on HF values?
Yes
Compute the HF value for every row and append it to the table
No
Arithmetic filtering operator?
Yes
Sort the table in descending order of a given column
No
Filter the table according to the filtering expression
Whole row required?
No
Strip the table keeping only the requested values
Yes
Global filtering enabled?
Yes
Retrieve & decrypt the results stored so far
Merge the previous with the current results
No
Boolean expression included?
Yes
No
Encryption enabled?
Yes
Encrypt all the results
No
Encapsulate all the results
Polling mode?
Broadcast
Sleep for a polling interval
Segmentation
Migrate to the next host

Figure 5. SNMP table filtering operation flow diagram

In addition to the scenarios already examined, the construction of filtering expressions with increased complexity has been also considered. Thus, MAs may be constructed with the ability to apply arbitrarily complex boolean expressions, namely logical *AND* and *OR* operators correlating individual filtering functions.

An example employing the *AND* operator would be: "*return the interfaces with utilisation 0.6<U(t)<0.8*". The output array *bigger* of the *Bigger* method invocation (mentioned in a previous example) would then be passed as a parameter to the *Less* operator to apply a second level of filtering (see Figure 5). In contrast, when the *OR* operator is considered, the two output tables (arrays) resulting from the individual expressions are simply concatenated.

V. CONCLUSIONS

This paper introduces the concept of *adaptive* hierarchical management enabled through MDM entities 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 [9][18]), 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. The framework is also designed so as to provide more balanced use of memory and processing power as MDMs always choose to execute at the least loaded host.

Management scalability is also further improved as apart from employing mobile mid-level managers, MDMs themselves rely on other MAs for data collection where filtering opera-tions are applied locally. In this context, we have proposed three applications of this framework in network monitoring, where MAs have been utilised to: (i) aggregate several MIB values into more meaningful network health indicators, (ii) acquire SNMP table snapshots, and (iii) filter SNMP table contents applying complex filtering expressions.

REFERENCES

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

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

[3] W. Farmer, J. Guttman, and V. Swarup, "Security for mobile agents: Issues and requirements", Proc. 19th National Information Systems Security Conference, pp. 591-597, Oct. 1996.

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

[5] D. Gavalas, D. Greenwood, M. Ghanbari and M. O'Mahony, "Complementary Polling Modes for Network Performance Management Employing Mobile Agents", Proc. IEEE Global Communications Conf. (Globecom'99), pp. 401-405, Dec. 1999.

[6] GMD Fokus, IBM Corp., "The OMG MASIF Standard".

[7] G. Goldszmidt, Y. Yemini and S. Yemini, "Network Management by Delegation", Proc. 2nd Int. Symposium on Integrated Network Management (ISINM'91), April 1991.

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

[9] A. Liotta, G. Knight and G. Pavlou, "On the Performance and Scalability of Decentralised Monitoring Using Mobile Agents", Proc. 10th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM'99), pp. 3-18, Oct. 1999.

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

[11] R. Pinheiro, A. Poylisher and H. Caldwell, "Mobile Agents for Aggregation of Network Management Data", Proc. Joint Symposium: 1st Int. Symposium on Agent Systems and Applications / 3rd Int. Workshop on Mobile Agents (ASA/MA'99), Oct. 1999.

[12] A. Pualiafito, O. Tomarchio and L. Vita, "MAP: Design and Implementation of a Mobile Agents Platform", Journal of Systems Architecture, 46(2), pp.145-162, Jan. 2000.

[13] A. Sahai and C. Morin, "Enabling a Mobile Network Manager through Mobile Agents", Proc. 2nd Int. Workshop on Mobile Agents (MA'98), LNCS vol. 1477, pp. 249-260, September 1998.

[14] SNMP research, "The Mid-Level Manager", http://www.snmp.com/products/mlm.html.

[15] Sprenkels R. and Martin-Flatin J.P., "Bulk Transfers of MIB Data", The Simple Times, 7(1):1-7, 1999, http://www.simple-times.org/.

[16] W. Stallings, "SNMP, SNMPv2, SNMPv3 and RMON 1 and 2", 3rd ed., Addison Wesley, 1999.

[17] G. Susilo, A. Bieszczad and B. Pagurek, "Infrastructure for Advanced Network Management based on Mobile Code", Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS'98), pp. 322-333, Feb. 1998.

[18] M. Zapf, K. Herrmann and K. Geihs, "Decentralized SNMP Management with Mobile Agents", Proc. 6th IFIP/IEEE Int. Symposium on Integrated Network Management (IM'99), pp. 623-635, May 1999.