# Meta-management of Dynamic Distributed Network Managers (MEMAD)

Ran Giladi[1,2] and Merav Gat[2]

[1]InfoCyclone Inc.
ran@infocyclone.com
[2]Communication Systems Engineering Department
Ben-Gurion University of the Negev, Beer-Sheva 89105, Israel
ran@bgumail.bgu.ac.il

**Abstract.** Distributed network management systems (NMS) have become a crucial necessity, especially for overcoming centralized NMS restrictions such as scalability and inefficient use of network resources. Such systems will also be instrumental in meeting the need for high-power computers and storage capabilities on the NMS platform. Modern technologies used in distributed NMS include management by delegating agents and mobile codes. These methods lead to the creation of a hierarchical architecture, since it simplifies management of the distributed agents. Peer management results in a dynamic, survivable and efficient way of managing the network, but it requires a complicated metamanagement mechanism to handle the managers. This study suggests an architecture for this purpose. We term this model Meta-Management of dynamic Distributed network managers (MEMAD). The purpose of MEMAD is to enable Peered Distributed Managers (PDMs) to manage the network by executing delegated or predetermined common management tasks. MEMAD defines a small, shared, replicated, and partitioned database as well as inter-communication SNMP based primitives for providing PDMs with the ability to cooperate efficiently in managing the network.

## 1 Introduction

Most network management systems available today are based on a centralized network management architecture. This simple architecture is composed of two entities: the Network Element (NE) which contains an Agent (a server with a small footprint), that holds management information concerning its managed NE, and a Manager (a client that runs applications). The Manager provides the management applications with a network view while it manages the agents by polling and processing their information. The agent and the manager communicate by using the standard simple network management protocol (SNMP) [1]. The manager performs most of the management work – polling, processing and controlling the Network Elements (NEs), while the agents access the NE information base in order to retrieve or update management information.

A. Ambler, S.B. Calo, and G. Kar (Eds.): DSOM 2000, LNCS 1960, pp. 119 - 131, 2000.

In recent times, Web-based management systems have begun to use a technology that somewhat differs from SNMP systems in that the management station communicates with its users or applications via Web-based GUI (e.g., HTML, XML) [13,15]. Web-based technology also allows Network Elements to respond with HTML pages instead of with SNMP, CMIP, or DMI messages to the management station [14].

In large and complex networks, the managers have to control a vast number of agents and perform all the information processing to the extent that the managers are loaded with information coming from all the agents. As a result, the managers' limited resources and network accesses bottleneck the management system. Additional problems such as increased network congestion, a high probability of disconnecting agents from the manager, slow responses from managers, and lack of redundancy have thwarted the centralized network management architecture from efficiently managing today's networks.

Of late, we have begun to witness a new approach to network management that, in principle, decentralizes network management, or better yet distributes it. The distributed network management approach deals mostly with the distribution of network management applications by delegating management tasks to other management entities for execution. Modern technologies presently used in this kind of distributed network management are usually hierarchically structured, since management of the distributed agents is simple and inherent in such structures.

Management by delegation (MbD) was among the first mechanisms to be proposed for hierarchical network management [2]. By delegating management tasks to specific agents, the manager's load is decreased and the tasks are executed close to the Network Elements (NE). This leads to a decrease in management traffic, higher availability of the NEs, and less dependency on the manager. However, for executing management tasks delegated by a manager, it has become necessary to devise a flexible and generic mechanism to serve as an extension to the agent.

Another approach to hierarchical network management [3] introduced the SubManager as a dual function entity. This manager delegates management tasks to the SubManager with access authorization to the relevant NEs and their agents. The SubManager then executes the task while polling the agents for the required information. This hierarchical model uses an SNMP framework and extends the Management by Delegation model by enlarging the scope of management of the dynamically extended entity (the SubManager).

Other approaches to distributed network management that have been suggested include the addition of RMON (remote monitoring) capabilities [4] to the executing entity [5], or the use of a Mid-Level-Manager and SNMP protocols. In the latter, the Mid-Level-Manger is provided with a new MIB (management information base) definition [6], which makes it possible to receive delegated management scripts and execute them on the NE agents, as well as other distributed management tasks.

Recent technologies, such as mobile code, have resulted in the Mobile Agent [7] approach. This technology implements the delegation mechanism in a slightly different manner, in that in order to achieve better network resource utilization, every management task is delegated and executed by several agents and servers. The use of recent technologies, such as mobile code, intelligent agents, etc. [8-10], provides more sophisticated and efficient distributed methods than the various hierarchical

methods commonly implemented (for example, peer distributed network management and fully distributed network management [11] as well as improved delegation mechanisms [12]).

Nevertheless, most of the afore-mentioned approaches lack a proper mechanism for controlling the distributed managers, especially in non-hierarchical distributed management systems and their derivatives. We call this problem the metamanagement problem because while the execution burden is shifted from the centralized manager to other mid-level management entities, the (human) manager still has to plan, configure and control the participating mid-level managers, the task distribution, and the execution process. These activities are essential for preventing redundancy, contradiction, and inconsistent management activities. For large scaled networks this task is virtually impossible if not automated.

In this work, we suggest a peered distributed network management model, which is based on a meta-management distributed architecture (MEMAD) that handles Peered Distributed Managers (PDMs). In this system, each PDM can implement a fully functional management application and execute its own management tasks. The PDM can also receive and execute a delegated task or a mobile agent from some management application.

In every distribution system, we are confronted with inherent difficulties and overhead regarding the management and the synchronization of the distributed elements. The suggested MEMAD architecture enables the PDMs to operate optimally and dynamically according to the network, to the PDMs states and to the applications, while utilizing minimum network resources. MEMAD deals with the metamanagement of the PDMs, rather than the managerial tasks the PDMs have to perform. Self-configuration, information and task distribution, backup, recovery, remote, and proxy operations of the PDM will be supported by MEMAD, regardless of the management activity the PDM is involved with for the management application. MEMAD does not deal with the distribution (by delegation, mobile agents or otherwise) of management applications; rather it guarantee that the management of the PDMs is carried out precisely, effectively, and efficiently.

Each PDM that uses MEMAD is an autonomous system that controls its own physical management domain. It communicates and cooperates with other PDMs, it responds to messages from management applications and PDMs, it learns, and it is proactive. The union of these physical management domains and the resulting response from the PDMs cover the entire network and the entire range of the NMS functions.

The organization of the paper is as follows: The following section presents MEMAD's architecture and the PDM model. The main algorithms are presented in section 3. Implementation and future work are described in section 4 and section 5 concludes the paper.

## 2 The MEMAD Architecture

The MEMAD architecture for peered distribution of network management deals with how PDMs cooperate to carry out network management tasks, regardless of whether

the tasks are predetermined or delegated according to need. Each PDM manages its Network Elements (e.g., via SNMP) and is responsible for doing so in some territory that is defined in terms of space or function. MEMAD ensures that all PDMs are synchronized and cover the desired network efficiently and without overlapping or holes. Although we implemented MEMAD to deal with network scope in terms of space, it can be implemented also in terms of functions.

### 2.1 Objectives

MEMAD was designed so that all legacy systems can be integrated with a consistent peered or fully distributed network management. In other words, it allows SNMP agents to participate in a scalable and efficient distributed management system without altering the agents. MEMAD's components and primitives are kept as simple as possible, thus enabling all kinds of distribution mechanisms (peered or otherwise, that are using mobile agents or delegated tasks) to function optimally (optimal in the sense of network traffic, response time, scalability, processing power, storage capabilities, etc.). MEMAD enables distributed network management to be implemented without tedious human planning and configuration, while maintaining load balancing and moderate network resource usage.

### 2.2 Components and Network Setup

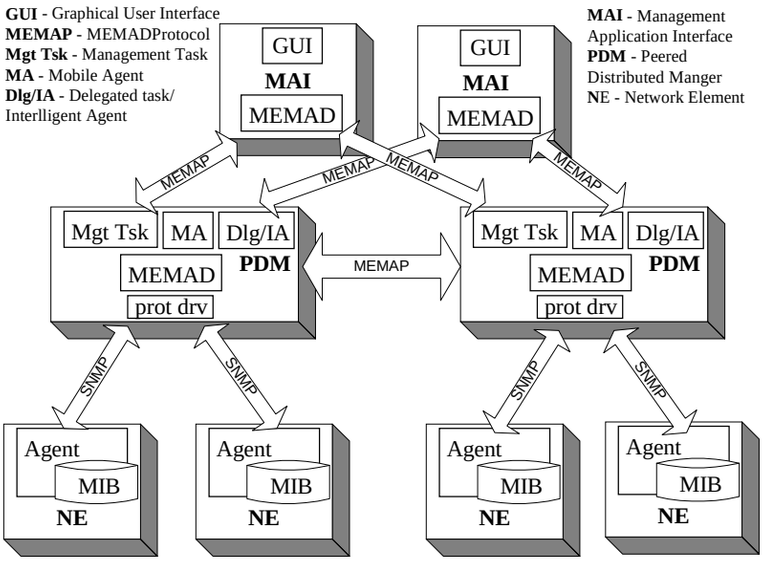


**Fig. 1.** Peer Distributed Management Entities

MEMAD defines a minimal set of entities as well as their capabilities (see Figure 1). They are described as follows:

- The *Management Application Interface (MAI)* is a very "thin client" which is separate from the manager and is used as an interface to the outside world. Through the management application interface and its GUI, a human manager can retrieve information and observe, hence, control the network.
- The *Peered Distributed Manager (PDM)* is the main entity of the model. This manager executes either a full management application or a delegated task on a limited group of Network Elements. The management application consists of a fixed set of management rules and tasks that enable the manager to perform independent activities on its Network Elements.
- The *Agent* remains within its traditional "thin server" architecture.

The PDM operates within the network where it manages the Network Elements (agents) for which it is responsible, and communicates with its peers and with the MAI addressing him, according to MEMAD definitions, as illustrated in Figure 2:

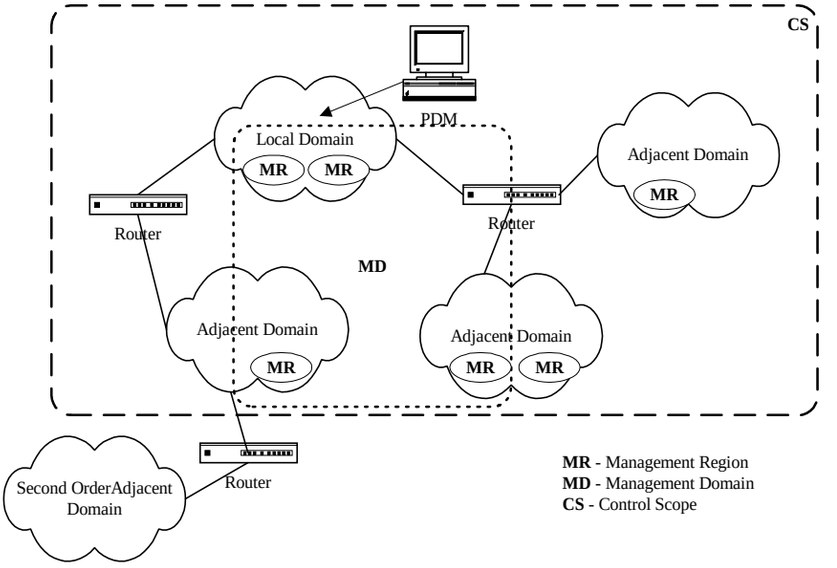


**Fig. 2.** Management Scope Definitions

- The *Domain* defines a sub-network (e.g., class B or C in IP networks). A *Local domain* of a PDM is the domain where this PDM is physically placed. An *Adjacent domain* of a PDM is a domain separated from the local domain by one router.
- The *Management region (MR)* is the collection of consecutive IP addresses within a specific domain. The *management region* of a PDM contains the agents to be managed by this PDM.

- The *Control scope (CS)* is the union of those domains in which a PDM can have any *management region*. In our work we restrict the control scope of a PDM to its *local domain* and all its *adjacent domains*.
- The *Management domain (MD)* of the PDM is its collection of *management regions* inside its control scope.
- The *Management weight (MW)* of a PDM is measured as the number of all the agents it manages.
- The *Local PDM* of a domain is the PDM that is physically placed in that domain, and the *Adjacent PDM* of a domain is the PDM placed in one of the adjacent domains.

We emphasize that the PDM has the sole responsibility for its management domain (unless it allows some other PDM to access its territory). However, several PDMs might reside in a sub-network (one domain). In that case, they will have to split the domain between them as equally as possible. There is also a situation when there is no PDM in a domain, and a PDM from another domain (usually an adjacent domain) "takes over" this domain as if it were its own. This "take over" can occur while the control scope is being reorganized, and will be carried out in as balanced a way as possible among all the PDMs in the system.

### 2.3 MEMAD Functions

In order to implement MEMAD we first had to construct a minimal set of functions. These functions include configuration, information distribution and aggregation (information regarding the network management itself), fault tolerance, and proxy management of various kinds, all of which are described below:

- The *Configuration* function enables a dynamic, adaptive and auto-configuration of the PDMs in the managed network, and includes primitives such as: "`hello`", "`I heard you`", "`who manage`", "`I manage`", "`acquire`", "`my world`", "`manager down`", "`unmanaged MR`" and "`re-managed MR`". Another set of primitives is concerned with load balancing of the PDM configuration and includes: "`balance`", "`balance information`", "`balance start`", "`balance start information`", "`balance end`", "`update`", "`update others`", and "`update split`".
- *Management information distribution and aggregation* include primitives such as: "`query`", "`sub-query`", "`answer`", and "`sub-answer`".
- *Fault tolerance* is a function that maintains a constant backup of the essential information and allows uninterrupted management and fast recovery. It includes choosing the right PDM to be the "hot backup" PDM, handling a partial replication of the metamanagement database in the network, and recovery processes. The extent of the replication is kept to a minimum so the network traffic stays low, while ensuring continuous operation and rapid recovery.
- *Proxy management* is a function that allows the PDM to share management tasks with other PDMs in the managed domain, or to allow other, remote PDMs to manage the domain. Primitives supporting this function include: "`remote`", "`share`", "`data`", "`data OK`", "`lengthen`", "`lengthen OK`", etc. These

primitives contain permission mechanisms that assure the security, period and scope of proxy management.

### 2.4 The PDM Database

Each PDM holds a management database that enables its management activities to be executed. The PDM has to manage its management domain in cooperation with other PDMs in the system at the same time that it performs its own distributed network management tasks. The PDM's database can be implemented using MIB tables (as we did in this study) or it can use other implementations.

The database contains a full description of the management domain and the control scope of the PDM. A description of other PDMs and their domains are sufficiently detailed in the database so as to enable MEMAD to operate.

The database is composed of three categories of information:

- A Class hierarchy that describes all the entities in the PDM's environment (device, link, node, role, etc.).
- A Collection of all the entities currently active in the PDM's environment and their detailed attributes.
- Detailed management information about the PDM's control scope: management regions, the entities residing in these regions, other active peers, management domains and management states, etc.

When the PDM is initiated, its database contains only its class hierarchy. In the process of identifying other peer managers (even before executing the "load balancing" algorithm), the PDM polls them for the necessary information about their management domains. Once the PDM finishes the initial data acquisition process, it updates the database with the relevant changes that occurred in its environment.

### 2.5 MEMAD Protocol (MEMAP)

The MEMAD protocol (MEMAP) was devised to enable PDMs to communicate both with each other and with the MAIs. The MEMAP is applied between the MEMAD layers in each PDM or MAI participating in the distributed management. We implemented the database using an MIB structure, and used SNMP queries to tunnel the MEMAP, although other methods are possible (e.g., CORBA, WEBM, etc.).

The MEMAP consists of a set of messages that enable primitives such as those described in section 2.3 to be transferred.

### 2.6 MEMAD Procedures

MEMAD ensures that all its participating components are able to provide a coherent and consistent view of the network, and that they share the managerial tasks equally. For this, MEMAD entities must inter-communicate and operate in the following way:

A user (or management application) approaches one of the MAIs for some management task (get or set some attribute in the network, analyze or produce collective information, etc.). This MAI uses its MEMAD layer to reach a default PDM (usually one that was placed in the same domain), and hands over the management task to this PDM.

This PDM becomes the "organizer" PDM, and activates a process, which we call the Integrator process. This process uses its PDM MEMAD layer to find the relevant PDMs for executing the required management tasks by checking the required tasks (e.g., IP addresses) with the management domains of all the PDMs listed in its PDM's database. The organizer PDM then retrieves the information, sets attributes, or initiates management tasks in these PDMs via their MEMAD layers. The PDMs operate with their Network Elements via their common interface, SNMP, CMIP, DMI, etc. The responses are then collected by the organizer PDM, aggregated (see, for example, [13]), and sent to the querying MAI.

In order to locate the relevant PDMs for the various managerial tasks, the PDMs must be configured in a way that makes them recognizable to all MEMAD participants. This configuration is a dynamic process, and is carried out by the PDMs themselves. The PDMs divide the network among themselves by assigning parts of the network to each of them as their management domains. Whenever there is a change in a PDM's status (e.g., a disconnection, a failure, etc.), the managed Network Elements are regrouped and assigned again to the PDMs, so that the management weight of the PDMs is distributed equally. A description of the underlying algorithms is provided in section 3.1.

Initially, there are no PDMs present, and no management activities. Any PDM joining the system defines its management domain. The first PDM to appear takes over the whole network. Other new PDMs will check with the active PDMs and build their own group of Network Elements and management domains, thereby decreasing the management weight from their peers. This procedure of reorganization is carried out using a load-balancing algorithm that will be explained in section 3.2.

This self-configuration, i.e., the dynamic definition of the management domains, also takes place when a PDM fails (while its backing PDM initiates the reorganization procedure), or on a periodical basis initiated by a local PDM.

Other procedures support the PDMs in maintaining a fault tolerance system, ease of operation and other special requirements. These include backup and recovery processes, remote management, and shared management processes. The backup management enables every PDM to be backed-up by one of its peers (see section 3.3).

The remote management procedure allows a temporary connection between two PDMs to be established. It enables a PDM to take a respite from management activities without being declared as failed, and it allows a remote PDM to take control over its management domain. A PDM that needs time off finds another PDM that agrees to serve as its remote manager for a predetermined period of time. During the remote connection, the remote manager takes responsibility for all management activities in the management domain of its connected PDM. When the remote connection ends, the original PDM returns to manage its own (updated) management domain. Another motivation for this kind of proxy management results from the need

of a PDM to watch some domain in a detailed resolution which is required by its management application. In this case, the remote manager initiates the procedure.

The sharing management procedure enables yet another type of temporary connection to be created whose purpose is to lighten the management's burden. This PDM looks for a peer that will agree to share its management domain for this period of time. During the shared session the volunteer PDM is responsible for all the management activities of the managed Network Elements that it has received from the initial PDM. Upon ending the connection, the initial PDM fully remanages its management domain.

# 3 MEMAD Algorithms

This section presents some of the main algorithms of the MEMAD architecture.

## 3.1 PDM Initiation

The initiated PDM's database is empty, thus the PDM is not aware of any other PDMs or agents in the system. Neither is he aware of the control scope nor of any other domain. There is no "super-manager" from which the new PDM can get information, nor is there a "super-manager" to introduce the initiated PDM to other PDMs. Consequently, the PDM has to search and learn the control scope by itself and then introduce itself to its peers within this control scope. During this "introduction" the PDM can seek information from other PDMs so as to fill his database. The PDM initiates a load balance algorithm after it learns the control scope, and then becomes an active PDM, that is equal to its peers.

To learn about its control scope, the initializing PDM performs a topology analysis. After the control scope is known, the PDM discovers other PDMs by broadcasting introduction messages to all domains within its control scope. Every PDM that receives an introduction message acknowledges it with information on the control scope and on other PDMs in its local or adjacent domains. In other words, the initializing PDM discovers another PDMs when it receives information regarding their network view. Both the initializing and the discovered PDMs are synchronized, and the initializing PDM synchronizes also with the PDMs he learnt about from the discovered PDMs. This method enables scalability, because it is not necessary to reach all PDMs while initializing, and due to the asynchronous way the PDMs are discovered and updated.

## 3.2 Load-Balance Algorithm

This algorithm deals with the load-balance between the participating PDMs and the distribution of management domains among the PDM within the control scope. Load balancing means distributing the load of management activities as evenly as possible between the managers, but it can be based on any optimal criteria. Examples might

include traffic, complexity of management tasks, a PDM load or response time, quantity of Network Elements, physical distance from Network Elements, network resources or their utilization, significance of Network Elements, reliability or quality of links and subnetworks, or a combination of these criteria. We implemented a simple objective function that is based on the quantity of the managed Network Elements and their network distance from the PDM, according to which the load-balance was computed. Thus, balancing the load in our implementation means that there exists an equal quantity of managed Network Elements per PDM in the vicinity of the PDM.

The load-balance algorithm is used when a new PDM enters the system or when a PDM terminates (gracefully or not), or on a periodical basis.

The algorithm is executed under the following constraints:

- Only one PDM at a time, within a certain control scope, can execute the load-balance algorithm. For the execution of load-balance it is necessary that all communicating PDMs be synchronized (non-communicating PDMs are not relevant).
- A PDM will always first be assigned to its local domain. It might also be assigned to other domains in its control scope, if they don't have any local PDMs.
- Management domains cannot overlap.

The load-balance algorithm consists of two consecutive stages: balancing the management load in the local domain, and balancing the management load in other adjacent domains that contain no local PDMs.

These two stages of the load-balance algorithms are almost identical and the only reason for the different balancing processes is the network distance criteria. According to the objective function, a PDM should manage the closest Network Elements it can locate. For this reason, the PDM should first balance the load in its own local domain and only then balance the load in the other domains.

To fully synchronize the load-balance algorithm, the PDM that executes the load-balance algorithm gets updated information from all its peers. Each PDM delivers information about its management regions, from which the balancing PDM can derive the management weight (MW) and the average distance attributes of the sending PDM. Additional data assembled by the PDM that executes the load-balance algorithm includes:

- Its distance from each management domain.
- The distance ratio (DR) of each management region (MR).
  *DR = distance of MR from its original (local) PDM / distance of the MR from the balancing PDM.*
- OMW - Optimal Management Weight (MW) in the domain:
  *OMW = MW / number of PDMs in the domain.*

Once all the required information is available, the PDM that executes the load-balance algorithm simply attaches (logically) the MRs with the lowest DR to its MD, until its MW reaches the OMW point. Afterwards, it balances the other PDM's load, but with one difference in the algorithm, namely, there is no DR parameter. In this case, the balancing PDM will try to remove an MR from one PDM and assign another PDM to

it if the MR remains intact. The second stage (balancing the adjacent domains) is performed exactly as done for balancing the other PDM load in the first stage.

The load-balance algorithm is terminated when the balancing PDM notifies so to all its peers, and when it informs them of all the changes in their management domain (MD). Once a PDM receives a message of a changed MD, its responsibility is to actuate these changes by sending a relevant message to the other PDMs. These PDMs become the new managers of its prior management regions (MRs).

The synchronization of executing the load-balance algorithm among the PDMs in the CS is achieved by using a "balancing queue" which is synchronized and maintained by all PDMs. This queue lists all those PDMs who are waiting to execute the load-balance algorithm. A PDM can execute its load-balance algorithm only when it reaches the top of the queue.

### 3.3 Backup Management

The backup algorithm is required in order to increase MEMAD's reliability. It dictates that every PDM has another PDM backing it up. When a new PDM gets its MD it has to look for a PDM to back it up. The most suitable PDM is chosen according to a set of parameters: the shortest distance between the PDMs, the lowest MW, and the absence of any other backup responsibilities. The new PDM polls all its peers for the relevant information and only when it finds the right candidate does it initiate the backup request.

Once the backup connection is made, the backed-up PDM updates its backup PDM on a periodical basis. These update messages contain the necessary data to enable the backed-up PDM to recover and reenter the system without starting the full process of an initiating PDM.

If the periodical update is not received on time, or in the case when a "manager down" message is received from another PDM, regarding the backed-up PDM, the backup PDM will know that the backed-up PDM is faulty. The backup PDM will then try to establish a connection with its backed-up PDM, and if it fails, the backup PDM takes full responsibility for the backed-up PDM's MD.

## 4 Implementation and Future Work

We are currently implementing MEMAD on 30 small sub-networks, using Java and JMAPI. Screening the networks, learning its topology and locating SNMP agents is performed with Win-Socket32, while communicating with the SNMP agents is done with JMAPI. The communication between the PDMs is carried out using Java-RMI. A combination of an SNMP-like protocol and RMI was created for maintaining an SNMP-based MIB to serve as the PDM's database.

The management task we tested dealt with the auto-discovery of network topology, which is carried out by several PDMs in parallel. We intend to extend this implementation for executing vast accounting management applications that call for

continuous sampling of equipment in various places of the network, storage of huge amounts of data, and analysis of aggregated information.

## 5 Conclusions

This work concentrated on the metamanagement problem of distributed network management systems (MEMAD). We described the essential features of such systems and designed and implemented the necessary algorithms for carrying out these features. Finally, the system was tested by an application for network auto-discovery.

The results showed that the distributed system configured itself very quickly, and was able to adjust itself to all changes in PDM availability, network fluctuations, and management application requirements. MEMAD was implemented for minimum distributed network management tasks, allowing all kinds of distributed mechanism to be employed. MEMAD can be implemented by any distributed network mechanism (e.g., hierarchical, delegated management tasks, intelligent agents, mobile codes, etc.). Finally, MEMAD exhibited a dynamic auto-configuration, fault-tolerance and responsiveness to management tasks.

Although we implemented MEMAD to deal with network scope in terms of space, it can be implemented in terms of functions as well.

## References

1. Subramanian, M.: Network Management: Principles and Practice, Addison-Wesley, Reading, MA (2000)
2. Goldszmidt, G., Yemini, Y.: Distributed Management by Delegation. In: the 15th International Conference on Distributed Computing Systems, Vancouver, British Columbia (1995)
3. Siegl M., Trausmuth G.: Hierarchical Network Management: a Concept and its Prototype in SNMPv2. Computer Networks and ISDN Systems **128** (1996) 441-452
4. Waldbusser S.: Remote Network Monitoring Management Information Base. RFC 1757 (1995)
5. Williamson B., Farrell C.: Distributed Management using Remote Monitoring Management Information Base (RMON MIB) and Extensible Agents. Technical Report, Curtin University of Technology (1996)
6. IETF Distributed Management Working Group (disman), RFC2591-2 and various drafts (http://www.ietf.org/html.charters/disman-charter.html) (2000)
7. Baldi, M., Gai, S., Picco, G.: Exploiting Code Mobility in Decentralized and Flexible Network Management. In Proceedings of the 1st International Workshop on Mobile Agents (MA'97), Berlin, Germany (1997)
8. Cheikhrouhou, M., Conti, P., Labetoulle, J., Marcus, K.: Intelligent Agents for Network Management: a Fault Detection Experiment. In Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management (IM'99), Boston, MA (1999) 595-609
9. El-Darieby, M., Bieszczad, A.: Intelligent Mobile Agents: Towards Network Fault Management Automation. In Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management (IM'99), Boston, MA (1999) 611-622

10. Zapf, M., Herrmann, K., Geihs, K.: Decentralized SNMP Management with Mobile Agents. In Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management (IM'99), Boston, MA (1999) 623-635
11. Sahai, A., Morin, C.: Towards Distributed and Dynamic Network Management. In Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'98), New Orleans, Louisiana (1998)
12. Breitgand, D., Dolev, D., Shaviner, G.: HAMSA: Highly Available Management System Architecture, Technical Report, Hebrew University of Jerusalem (1999)
13. Anerousis, N.: A Distributed Computing Environment for Building Scalable Management Services. In Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management (IM'99), Boston, MA (1999) 547-562
14. Martin-Flatin, J.P.: Push vs. Pull in Web-Based Network Management. In Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management (IM'99), Boston, MA (1999) 3-18
15. DTMF standards concerning DMI, WBEM, DEN, http://www.dmtf.org, 1998-2000.