On the use of Mobility in Distributed Network Management

Rui Pedro Lopes(1), José Luís Oliveira(2)
(1)*Instituto Politécnico de Bragança, ESTiG, P-5300 Bragança*
(2)*Universidade de Aveiro, DET, P-3810 Aveiro*
rlopes@ipb.pt, jlo@ua.pt

Abstract

Information Technology has been under unprecedented transformations and it is dramatically changing the way of work inside organizations. Information management systems must be adequate to cope with the profound effects of this evolution, which expectations includes the introduction into the networks of enormous quantities of different elements. Mobile agent paradigm seems to be, for many researchers, the right solution to deal with the pressures of these new demands.

This paper discuss the issues around mobility of code on network management environments and presents ongoing work that provides mobility capability to distributed managers upon recent work of IETF's Disman working group.

1. Introduction

Internet technologies and concepts are dramatically changing the way enterprises provide, maintain and use traditional IT services. Management systems have to accomplish the profound effects of this evolution that will introduce into the Net enormous quantities of different elements ranging from low resources devices to large-scale distributed applications. New paradigms must be available to deal with the pressures of these new demands. Many researchers believe that mobile agent paradigm can provide one of such answers.

The term Mobile Agent refers to autonomous programs that can move across networks, from node to node, and assume agent behavior, i.e. act on behalf of users or other entities [1]. Most of the research efforts on this topic have been developed under the context of the telecommunications market. Examples of such research have been: management distribution and delegation [2], network services deliver [3] network traffic optimization and network's fault tolerance [4].

During the past decade the network management market was largely dominated by the IETF's Management Framework (shortly SNMP). However this model has suffered from several drawbacks during its evolution. Even the recently proposed draft standards of SNMPv3 (April 1999) present a set of lacks that avoids the model to cope well with the new near-coming demands of bandwidth, equipment and services.

Some management operations require large bandwidth from the network, due to the necessity to transmit, for instance, tabular information. Management protocols are not well designed to deal efficiently with this type of transfer load [5]. By changing the paradigm "move data" to "move code" mobile agents seems to save precious network resources. But is it always an advantage to use mobile agents? Several issues and scenarios have to be analyzed in order to infer about that. We just point a few:

- Enterprise organization – Management processes tend to mirror the enterprise organizational structure. Most enterprises are divided over several regional agencies that depend upon a central system to perform management operations. On these scenarios mobility of code may help reduce long lines permanent traffic that is typically associated with request-response management approaches.
- Management operations – The type of primitives that is requested also has impact on the efficiency of mobile agents. The processing of huge MIB data is more efficient if performed locally, moving code instead of data. On the other hand, the configuration of some equipment parameters can be more cost-effective if performed on client/server transactions.
- Historical knowledge – The agent must keep knowledge about previous operations in order to improve its behavior over the time.
- Network resources – The agent mobility have to be support by a special infrastructure, which must be present beyond the already provided communication framework. Moreover, agents should consider moving when the estimated efficiency of the next migration node is better than the performance of the current location.

Other research topics around the design and development of mobile agents architectures and its applications, covers currently a large spectrum ranging from security and privacy to the legality and ethics [1].

There is today a large debate on the use, or not, of mobile. A lesson that was already learned concerning mobility is that the technology is available (and will be even more). The market will decide when and how to use it, depending on several factors such as performance, security, development cost and end-user requirements. We will show an example of one such application.

This paper investigates and promotes the inclusion of mobility in distributed management, using, as a base, the recent work of IETF's Disman working group.

2. The SNMP management framework

Network management issues were dominated over the last decade by two main approaches: the OSI Management Framework or shortly the CMIP/GDMO model and the IETF Management Framework (typically identified as SNMP). Due to its simplicity SNMP soon gained a larger set of followers relegating the CMIP to a second and also insignificant market share (due to its powerful capabilities, some supporters still remain specially across telecommunication operators).

The IETF network management framework is based on a reduced set of concepts. Considering the traditional naming of this model, which is in use since the SNMPv1, key elements are:

- Agents or managed nodes – conceptually these agents represent network elements, such as a bridges, router, switch, printers, etc., that collects information on behalf of management purposes (management information).
- Manager (or Network Management System, NMS) – typically provides the end-user with a set of management operations performed across the management information spread over the agents.
- Management Information Base (MIB) – represents, in a directory-like structure, the management information that can be handle in each Agent.
- Management Protocol - SNMP - the standard way to exchange management information between systems (manager-agent).
- Proxy Agent - allows the integration on the SNMP management framework of "dummy" equipment that does not include SNMP services.

The SNMP framework defines both the structure and attributes used to face with managed resources (SMI and MIB), and the way these resources are retrieved (management protocol - SNMP). It does not define upper level tasks such as interpretation, correlation, and corrective measures. This type of decision must be performed by specific management applications.

2.1. SNMPv3

SNMPv3 try to eliminate previous versions weaknesses by the inclusion of some new features. Among these are the security support and a flexible architecture that allows the redefinition of current modules or the introduction of new parts inside the framework. Each SNMP configuration is classified as a "SNMP Entity" composed by several interacting modules: Dispatcher, Message Processing, Security, Access Control and Application module. The combination of these modules allows providing different SNMP roles (i.e. an agent, proxy or manager) [6].

The Application(s) use services from the SNMPv3 engine to send and receive messages, authenticate, encrypt and control the access to managed objects [7].

The Dispatcher subsystem coordinates the communication between SNMPv3 engine subsystems and differentiates modules belonging to the same subsystem. Based on the PDU information, it determines which application should be invoked and coordinates the respective transport mappings.

In a working scenario, before transmitting the message, the dispatcher checks the selected protocol version and type (Figure 1). Following this information, the adequate message processor is invocated which itself relies on the next subsystem (security) to pack the message.

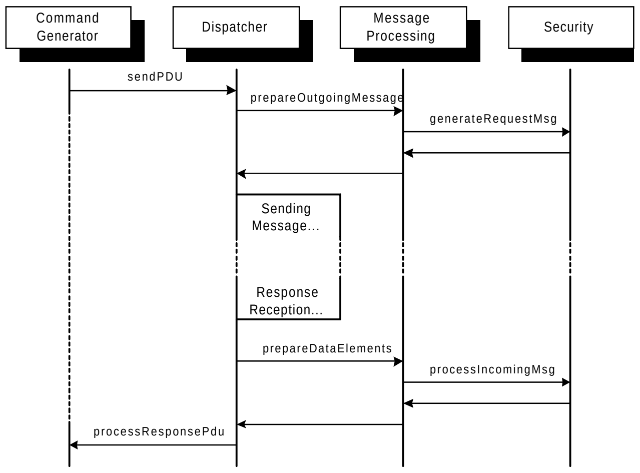

Figure 1 – Command generation inside an SNMP Entity.

2.2. SNMP evaluation

The SNMP framework is a centralized approach, i.e. a NMS uses distributed agents to collect management information. This data is retrieved on demand by the NMS to be processed.

This approach has some drawbacks, in particular due to the lack of extensibility and scalability of the model on very large networks. This constraint results from the inability of a centralized manager to handle huge amounts of management information and also because centralized polling across geographically distributed sites is infeasible and expensive [5]. Moreover, system updates usually entail the modification of several agents or of the management station itself. In addition, there are occasions where it is necessary to cope with situations where the management station is not accessible. The classic management architectures are not well suited for low-bandwidth or disconnected operation.

Several authors have addressed these problems along the past years [2] [8] [9] resulting in ad-hoc and partial solutions typically based on management distribution and delegation. Inside the IETF, the Distributed Management (Disman) WG was chartered to define an architecture where a main manager can delegate control above several distributed management stations thus improving scalability through distribution and allowing “off-line” operations.

2.3. Disman

The management distribution allows reducing the processing load on the traditional centralized management station (NMS) by delegation tasks upon several Distributed Managers (DM) or upon more powerful agents. A DM is an SNMP entity that receives requests from another manager and executes those requests by performing management operations on agents or other managers.

Since the management entities are split over the network and collaborate between themselves by assignment, a hierarchy of several “islands” is created increasing the robustness and fault tolerance of the overall management system (Figure 2). Although if the access to the central manager is not possible, each DM may handle locally critical situations.

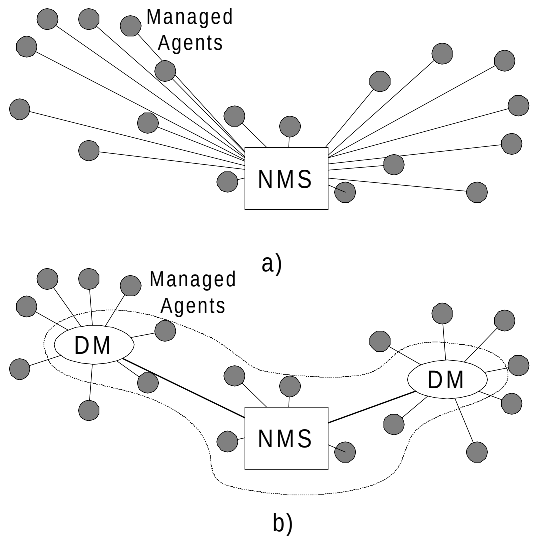

Figure 2 – A centralized management approach (a) versus a Disman-based one (b).

The IETF Disman framework is based on distributed management applications and services. A distributed management application performs some management function, often by monitoring and controlling managed elements. The distributed management services can perform functions or store information once for all applications on the local system thus making a set of applications more efficient. Each service is provided by a specific MIB interface.

Currently there are being proposed several MIB to address different but complementary issues of management operations distribution [10].

The Event MIB is the successor of the SNMPv2 Manager-to-Manager MIB. It provides the ability to monitor MIB objects either locally or remotely and takes an action when a trigger condition occurs.

The Expression MIB was developed to allow the definition of MIB objects that designers have not provided yet. In other words, it supports externally defined expressions of existing MIB objects. This allows the definition of expressions of expressions. The Expression MIB allows to provide the Event MIB with custom-defined objects. The result of an expression can trigger an event, resulting in an SNMP notification. Without the Expression MIB such monitoring is limited to the objects in predefined MIBs.

The Notification Log MIB is intended mainly for Notifications providers (traps or informs) but may be also used by consumers. It defines a mechanism to cope with Notification losses by recording (logging) Notification information.

Each of the Remote Operations MIBs (*ping*, *traceroute*, *lookup*) enables the remote correspondent operation to be performed at a remote host. It will provide a standard way to perform remote tests, to issue periodical sets of operations, and to generate Notifications with test results.

The Schedule MIB provides the definitions to perform the scheduling of actions periodically or at specific times and dates. The actions are modelled by SNMP set operations on local MIB variables (restricted to INTEGER type). More complex actions can be realized by triggering a management script, which is responsible for performing complex state transitions.

The Script MIB module allows the delegation of management functions to distributed managers. Management functions are defined as management scripts written in a language supported by the managers. It may be a scripting language (such as TCL) or native code, if the implementation is able to execute native code under its control. This module does not make any further assumptions on the language. The distributed manager may be decomposed in two blocks: the SNMP entity, which implements this MIB, and the runtime system, capable of executing the scripts. The Script MIB sees the runtime system as the managed resource, which is controlled by the MIB. The runtime system can be defined as an SNMP application, according to the SNMPv3 architecture.

2.4. Disman evaluation

Although the distribution of management tasks by several autonomous stationary agents, as proposed in the Disman framework, is an improvement when compared to centralized architectures, it also suffers from some problems:

- The communication protocol, SNMP, uses IP addresses to identify the peer. If there is a firewall between the two SNMP end-points, it may be used invalid IP addresses (192.168.0.xx, for example). In this case it is not possible to contact the peer unless some tunneling mechanism is used [11]. Moreover, the IP addressing scheme is too rigid in terms of peer location. Using higher level naming mechanisms makes it is possible to get references to objects inside or outside the firewall in a location independent way.
- System update usually entails the modification of several services and management applications most of them manpower based.
- Appending new features usually makes the management applications or the management services more complex. Furthermore it is not easy to dynamically introduce/remove features to increase performance.
- The stationary distributed agents may suffer processing overloads and no mechanism is available to cope with it [12].
- The correlation of management information retrieved from several DM is possible only through a hierarchical flow.

Nevertheless, Disman is the right move to handle the crescent complexity of networks management – “Divide to conquer”. Considering just the technological issues associated to Disman implementation we see here a great field where the mobile agent paradigm can prove it capability, with general enrichment of the architectural functionality.

3. Mobile agents

Far goes the time when computers were exclusive of teams of experts. Nowadays, a large and increasing number of users with diverse skills and different levels of knowledge use computers and network systems. In this scenario, where the information is the central issue, users usually feel a need for extra eyes, hands, time and even brainpower. Agent technology has been promoted as the way to make the difference. With the ability to perform actions on behalf of other programs or on behalf of the user, agents can be applied on areas such as information retrieval [13], electronic commerce [14], personal mobility, telecommunications and user interfaces [15] just to name a few. A mobile agent is a particular case of these software agents that have the ability to migrate from host to host.

Mobile network agents are pieces of code that can be dispatched from one computer and transported to a remote computer for execution. At the remote host, they present their credentials and obtain access to local services and data (maintaining the previous execution state). Each host (Agent System) may also serve as a broker by bringing together agents with similar interests and compatible goals, thus providing a meeting place at which agents can interact.

Several approaches have proposed mobile agents for managing networks and services, due to the distributed nature, efficiency savings, traffic reduction and robustness [16-18]. Besides the gain in flexibility and efficiency it also brings back some shortcomings, namely, it increases the agent management difficulty [19] and also introduces some kind of usability challenges and possibly threats [20].

3.1. Agents technology

Mobile agents are based on two distinct technological concepts: the agent aspect and the mobility facility. The mobility facility deals with aspects such as code and state mobility. The agency aspect is the “acting as an agent” part. In this section we present a generic model for the mobility concept.

Agents have a well define lifecycle composed of the states and transitions that it can perform (Figure 3):

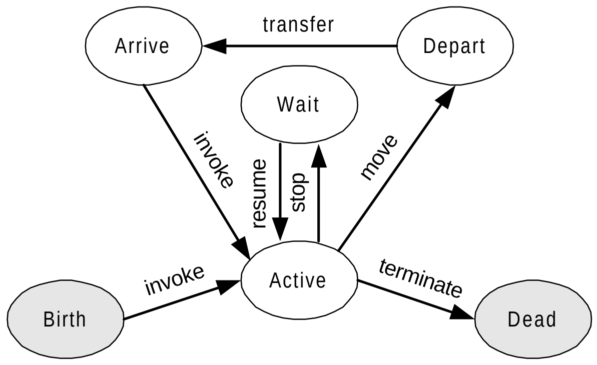

Figure 3 – The mobile agent life cycle.

- Invoke – performs setup activities such as building data structures and starts executing its code.
- Resume – restores intermediate results and continues executing its code.
- Stop – saves intermediate results and stops its execution.

- Move – stops its execution, saves intermediate results, and the code is serialized.
- Transfer – the serialized code is transported to the destination.
- Terminate – performs one-time termination activities.

Nowadays there are several technological solutions to support the development of mobile agent systems. All of them provide a platform to support the migration and execution of code (agent platform) and use scripting or interpreted languages (such as TCL or Java) to cope with heterogeneity.

To ensure interoperability between mobile agent agencies, the OMG (Object Management Group) promoted the MASIF specification [21] but at the present time it has not gained a significant adherence.

4. Optimizing delegation through mobility

Before addressing this section main issue it is important to establish a clear distinction between several concepts that use the same acronym (MA): Mobile Agents, Managed Agents and Management Agents.

A Mobile Agent is a program or process that uses some set of functions or methods (API) that allows it to transfer the code and the current state to other location. Mobile agents are usually executed in a specific environment (mobile agent platform), which also provides security features, cloning and copying capabilities. At this time, we choose to consider that a mobile agent is composed only by mobility aspect and leave the agency aspect for latter on. For now on we identify it as a *MobAg*.

SNMP Agents are entities responsible for collecting and storing management information local to the node (MIB modules) and exchange this information through a management protocol (SNMP engine). The simplicity of the agent allows its embedding in restricted environments, such as bridges or hubs. We will name it as MN (managed node).

A Management Agent is a program or process (agent) that provides an abstraction of network resources with the purpose of facilitating management operations. It usually acts on behalf of another program or application, performs instrumentation operations and executes management operations. Note that a management agent can implement some SNMP module and thus have characteristics of SNMP agent. From this point on we will use the term *ManAg* to name it.

The development of a network management application, namely tools to gather and process management information, is a difficult task due to the ever-changing technology and standards. Moreover, aiming the interoperability it is important to maintain compatibility between the successive standards.

In the last years we have observed the emergence of some proprietary network management solutions, such as Java Management API [22] and Web-based management [23]. Although these architectures were developed over some of the SNMP weaknesses (special upon the SNMPv2 defect) they did not find, yet, a wide acceptance on the network management community.

The IETF, with the Disman framework, addresses some of the architecture faults as a way to improve scalability, robustness and flexibility from the classical centralized architecture. A mobile agent approach can complement this framework by providing location transparency and facilitating some cumbersome aspects such as the SNMP engine installation, data correlation and others (see Table 1). A particular feature that is emphasized in our approach is the concept of the Mobile DM, i.e., a distributed manager with mobile properties.

4.1. Design requirements

The proposed architecture meets the following objectives:

- Standard conformance – encompass current management framework, particularly the Simple Network Management Protocol. The SNMP framework is the most known and implemented architecture for network management. For this reason it is important that the system provides an SNMP engine. The Disman architecture, in particular, solves some distribution problems.
- Openness – the management community did not find yet a consensus on the choice of the adequate network management model. There are several different solutions available, either proprietary or standard. Maintaining the system architecture as modular as possible allows the interoperability between different models.
- Migration services – the ability to move around is important for some management operations so it should not be disregarded. The architecture should provide a set of services to allow the migration and cloning of management agents, regardless of the implemented management models. For example, if some network section is having communication problems, it may be necessary to move the DM to a different location, where it can operate.
- Maximize flexibility – by providing a open architecture it is possible to add or change modules implementing different services with the same kernel. The agency in the *MobAg* paradigm is similar to an operating system, which can provide the same resources to different applications and services.
- Maximize usability – the increase of network and applications complexity is a reality and a never-

Table 1 – Problems solving both by Disman and MobAg approaches.

Task	Disman solution	*MobAg* solution
DM installation	The user must install each engine in every host.	Each engine may be distributed by agent's cloning.
Inter-agent and NMS Communication	SNMP, HTTP, FTP – based on IP.	Method invocation (either remote or local), KQML, etc. – independent of the network protocol.
Data processing	Each engine processes data from a hierarchical set of other engines or distributed managers.	Each agent may process data from several sources. The processing is horizontal.
Data correlation	Must be done hierarchically by a higher level DM based on data collected by SNMP engines or other DMs.	A unique agent may move from node to node correlating the previous collected information and results with the local information.
Heavy load processing tasks	It is difficult to implement some kind of task distribution mechanism.	Agent cloning followed by migration to a platform with low usage – better use of network processing resources.
Engine updates	Using script and expression MIB.	Code on demand. Agent customization followed by migration to the destination.
Network faults	Inside the DM "island" lacks on the network lead to unmanageable situations	A mobile DM can adapt its position according to traffic, processing targets and network faults.

ending process. Repetitive and boring management tasks should be performed automatically, without the user intervention. Moreover, the system should be permitted to take some decisions by itself (intelligent *MobAg*).

- Support for policies – according to different sections of the network it may be necessary to apply different policies. For example, in an accounting department, security management should be prioritized while in the multimedia section the management focus must be on bandwidth and QoS.

The following model tries to cope with the above requirements through the use of *MobAg* upon Disman architecture.

4.2. Mobile disman architecture

The proposed architecture uses *MobAg* to implement Disman specifically the DM entity. The mobility support in DMs allows them to adapt to a changing environment and simplifies tasks such as agent and tasks distribution.

Figure 4 presents two situations of DMs with mobile characteristics. In the first situation (one) the DM choose to clone to a different management domain because the instantaneous load increased. Situation two presents an approach where the communication with the upper management station is interrupted. As an example, the DM may have detected a problem in the platform where it was installed and choose to migrate to a different location to continue its operation without assistance from the management station. When the station gets back on-line it may migrate to the original host and continue its operation. Other situation where this move may occur is when the interaction between the DM and some agent delivers high volumes of traffic. In this case, instead of generating traffic across several links the DM can move near the agent and interact with it locally. The DM *MobAg* can dynamically infer about these condition in order to adapt to the best network position and the best host to perform.

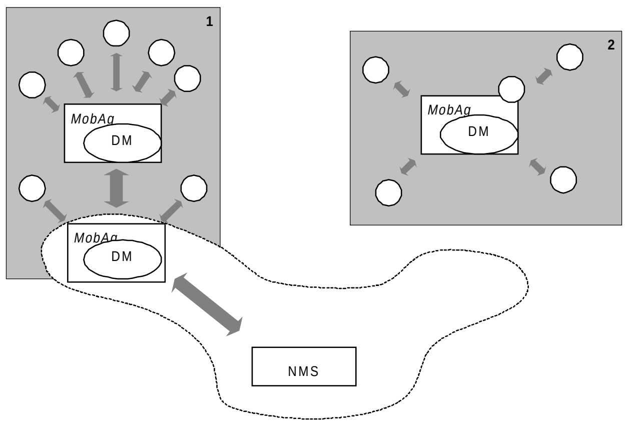

Figure 4 – Mobile disman architecture.

From the usability point of view, the user (manager) may define DMs in the topmost Management Station and set its behavior. After creating the desired DM he can define an itinerary to be followed or some kind of distribution policy.

For this architecture to work as expected it is necessary to define an architecture for the DM so that it can implement the Disman framework and also the mobility characteristic. It must be considered that the SNMP framework does not expect the agents to move, so it is necessary to maintain knowledge of the current location of the agents. This fact also helps managing the DMs. The join of the two entities implies the introduction of SNMP services inside an Agent System (the host kernel for agents).

4.3. DM implementation

The DM architecture was planned as open as possible but keeping in mind primary, the Disman conformance statements.

The main goal of this special agent is to collect raw data from several MNs, or from itself, and provide local management task and processing over that data. This gathering may be done by SNMP, method invocation (remotely or locally) or any other mechanism. By providing communication engines with a common external interface to communication services it is possible to inter-exchange modules maintaining the same architecture (Figure 5).

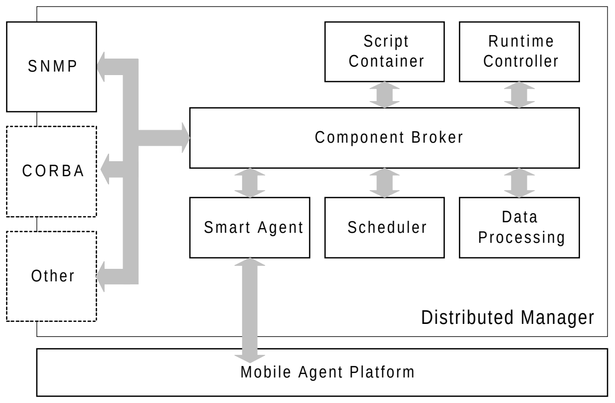

Figure 5 – Distributed manager scheme.

The instrumentation raw data must be filtered and processed in order to extract useful information. The *Data Processing* unit is based on the Expression MIB module and it collects algorithms, expressions and functions to process management information obtained from the managed nodes.

Management operations performed upon network components can be defined as scripts, according to the Script MIB module. The *Script Container* module maintains a collection of object references and it is used to carry the scripts and macros along when the DM migrates. The containers also provide a persistence service i.e. non-volatile storage. The *Runtime Controller* is responsible for recognizing the scripts language and version, so that it can choose the necessary interpreter. In addition it provides methods to initialize, start, stop, suspend, resume and get the status of the script execution.

The DM *MobAg* have some level of autonomy that allows it to make decisions, such as when to move, when to clone, move to where. This characteristic is implemented by the *Smart Agent* module, which provides algorithms and functions to help deciding according to results from the Data Processing unit. If, for instance, the conclusion is that the DM platform is under heavy load, an agent will be dispatched to continue processing a specific task on another less loaded platform, or it can create, through cloning, several DM that operate in a hierarchical way under this DM's control.

The *Scheduler* launches notifications to registered modules either periodically or at a given time, according to the defined schedule providing time operational information that helps to guide management tasks activation and repetition.

The *Component Broker* is a lightness communication bus can controls and interconnects the other DM modules.

Since the DM do have mobility and cloning capabilities it must report do the upper level DM, the NMS at the uppermost node, each of such operations it executes (mobility monitoring). By doing this it is possible to maintain under control all the collaborating DMs.

4.4. Estimating about mobility

How well distributed computing systems perform depends a great deal on an efficient usage of network resources. Load balancing consists on the balancing of the total workload among the various processors of the underlying systems, in this particular case among the various DMs of the network.

In a static approach the manager distributes the estimated load along several points. Future re-planning must pass through the user/manager evaluation, which means, "managing the management system".

Moreover, network management systems are intrinsically dynamic thus increasing the load-balancing problem. Considering moving load between processors we arrive to a not so simple task as have been shown by several authors [24-26]. This approach, however, uses balancing at the application level that, in spite of having much lower granularity, simplifies significantly the processing model (monitoring of traffic impact, estimation of "near" real-time management operations, output delays, etc.).

The information needed to perceive this is not easy to define and to obtain, and is currently being matter for further investigation. Several parameters are being considered mainly based on two resources: the host performance and the network throughput and reliability.

5. Conclusions and future work

The management of enterprise networks, i.e., to monitor and to act on network components, is a task that involves commonly the use of heavy and complex applications. This difficulty arises from the diversity of network solutions, rapid technological advances and the adoptions of different technical solutions. The SNMP network management model proposed by the IETF addresses the data collection and transport but does not specify higher-level management solutions.

This paper has presented an implementation of the Disman framework that is complemented with mobility support. This architecture allows the Distributed Manager to adapt to a changing management environment. It allows increasing management efficiency by reducing management traffic, providing a better use of network resources and enhancing flexibility.

The mobility characteristic raises several new issues in Disman DMs. At the moment we are mainly concerned with the performance of the DM in terms of processing load balancing and network efficiency. The next step to be performed is to get historical data from network nodes and perform a study on it aiming at extracting "when to move" information. An adequate sensing mechanism and correct decision mechanisms may increase the management system overall efficiency.

6. References

[1] V. Pham, A. Karmouch, "Mobile Software Agents: An Overview", IEEE Communications, Vol. 36, No. 7, July 1998, pp. 26-37.

[2] G. Goldszmidt, Y. Yemini, "Delegated Agents for Network Management", *IEEE Communications Magazine*, Vol. 36 No. 3, pgs. 66-71, March 1998.

[3] S. Krause, T. Magedanz, "Mobile Service Agents enabling Intelligence on Demand in Telecommunications", *Proc. IEEE GLOBCOM'96*, 1996.

[4] R. P. Lopes, J. L. Oliveira, "Software Agents in Network Management", *Proc. of the 1st International Conference on Enterprise Information Systems – ICEIS'99*, March 1999, Setubal, Portugal.

[5] R. Sprenkels, J-P Martin-Flatin, "Bulk Transfer of MIB Data", *The Simple Times*, Vol. 7, N. 1, March 1999.

[6] J. Case, R. Mundy, D. Partain, B. Stewart, RFC2570 (I), "Introduction to Version 3 of the Internet-standard Network Management Framework", April 1999.

[7] B. Wijnen, D. Harrington, R. Preshun, RFC2571 (DS), "An Architecture for Describing SNMP Management Frameworks", April 1999.

[8] J. Oliveira, *Arquitectura para Desenvolvimento e Integração de Aplicações de Gestão*, PhD Thesis, University of Aveiro, September 1995.

[9] A. Brites, *et al.*, "A Protocol-independent Notation for the Specification of Operations and Management Applications", *Proc. DSOM'94, Fifth IFIP/IEEE International Workshop on Distributed Systems: Operations & Management*, October 1994.

[10] Distributed Management (disman) Charter, http://www.ietf.org/html.charters /disman-charter.html

[11] Mobile-IP: Supporting Transparent Host Migration on the Internet, http://anchor.cs.binghamton.edu/~mobileip/LJ/

[12] O. Shehory, K. Sycara, P. Chalasani, S. Jha, "Agent Cloning: An Approach to Agent Mobility and Resource Allocation", *IEEE Communications Magazine*, Vol. 36, N. 7, July 1998.

[13] I. Nekrestyanov, T. O'Meara, A. Patel, E. Romanova, "Building Topic-Specific Collections with Intelligent Agents", *Proc. of the 6th International Conference on Intelligence in Services and Networks*, IS&N'99, Barcelona, Spain, April 1999.

[14] M. Gleizes, A. Léger, E. Athanassiou, P. Glize, "Self-Organization and Learning in MultiAgent Based Brokerage Services", *Proc. of the 6th International Conference on Intelligence in Services and Networks*, IS&N'99, Barcelona, Spain, April 1999.

[15] S. Green, L. Hurst, B. Nangle, P. Cunningham, F. Somers, R. Evans, *Software Agents: A review*, May 1997.

[16] Perpetuum Mobile Procura, http://www.sce.carleton.ca/ netmanage/ perpetum.shtml

[17] Collective Intelligence, http://www-iasc.enst-bretagne.fr/ PROJECTS/SWARM/

[18] A. Bieszczad, B. Pagurek, T. White, "Mobile Agents for Network Management", Carleton University, Canada, 1997.

[19] M. Breugst, S. Choy, "Management of Mobile Agent Based Services", *Proc. of the 6th International Conference on*

Intelligence in Services and Networks, IS&N'99, Barcelona, Spain, April 1999.
[20] E. Kaasinen, "Usability Challenges in Agent Based Services", *Proc. of the 6th International Conference on Intelligence in Services and Networks*, IS&N'99, Barcelona, Spain, April 1999.
[21] MASIF, *Mobile Agent System Interoperability Facilities*, http://www.omg.org/.
[22] Java™ Management Home Page, http://www.javasoft.com/products/ JavaManagement/index.html
[23] WBEM Initiative, http://www.dmtf.org/wbem/index.html
[24] B. Ghosh, S. Muthukrishnan, "Dynamic load balancing in parallel and distributed networks by random matchings", *in: SPAA '94. Proceedings of the 6th annual ACM symposium on Parallel algorithms and architectures*, pp. 226-235.
[25] A. Tantawi, D. Towsley, "Optimal static load balancing in distributed computer systems", *Journal of the ACM Vol. 32, No. 2, April 198*, pp. 445-465.
[26] S. Muthukrishnan, R. Rajaraman, "An Adversarial Model for Distributed Dynamic Load Balancing", *in: SPAA '98. Proceedings of the tenth annual ACM symposium on Parallel algorithms and architectures,* pp. 47-54.