AN APPROACH TO SELF-MANAGEMENT BASED ON AUTOMATIC DIAGNOSTICS

Rui Pedro Lopes[(1)], José Luís Oliveira[(2)]

[(1)]Instituto Politécnico de Bragança, Escola Superior de Tecnologia e Gestão, P-5300 Bragança
{rlopes@ipb.pt}
[(2)]Universidade de Aveiro, Dept. de Electrónica e Telecomunicações, P-3810 Aveiro
{jlo@inesca.pt}

Abstract

Management operations imply, most of the time, the implementation of prevention policies in order to avoid faults and to provide expeditious answers to those faults.

This paper present an architecture that allows to infer about network symptoms and to correlate these with well-known anomalies in order to provide the adequate diagnostic that will drive to the fault repair.

I. Introduction

Nowadays, a typical data network is inherently heterogeneous. Rapid technological changes, different technical solutions and the accumulation of legacy systems overload today networks with many different kinds of systems, protocols and languages. All of these issues increase the network management difficulty.

With several of such environments around the world there is a need for a management system compatible with every system, as mainframes, UNIX workstations and servers, PC systems equipped with diverse operating frameworks. The variety in interconnection media and protocols might be even greater: Ethernet, FDDI, ATM, TCP/IP, IPX/SPX, NetBIOS and many others.

Recently, a number of technologies capable of surviving over heterogeneity emerged, such as DCOM [1] and CORBA [2]. Moreover, recent machine independent languages, as Java, can relieve the burden typically associated with the developing of distributed applications. From the management perspective there are all a panoply of new challenges. The proliferation of normalised and proprietary MIBs arises some interrogations about the usefulness of all this data.

A first approach to simplify the management operations construction is to provide an adequate abstraction of the overall MIBs, spread along the network [3]. This information is viewed as a virtual database that can be accessed remotely through an SQL interface, Java applets or a Common Gateway Interface (CGI).

From the human manager viewpoint, management operations can be boring and inefficiently performed, such as scheduled tasks (operations activated periodically and occasionally planned to start at a specific time). Most of the management operations are often based on human effort, both for the collection of information and for the processing of results. It is desirable that the Network Management System (NMS) provides increasing levels of automation, as a way to improve the decision support for a human manager [4]. An important contribution can be done by high level management models and expert systems mechanism that are based on control algorithms capable of pro-active management operations [5].

On the following sessions it will be described a Network Management System - the PRONET - that allows to reduce the volume of management information delivered to the user manager, by the use of an abstraction layer over the management data, and that integrates proactive management features based on the symptom-diagnostic paradigm.

II. Heterogeneous Systems Management

Nowadays, network managers are confronted with a panoply of different systems, network components and network technology (Figure 1). Besides the difficulties to maintain a communication network in a correct operational state, the user/manager must have a solid information about all the equipment in the network. The vertiginously advances in this area make the manager task even hard to perform. He has to maintain continuously updated about technology advances.

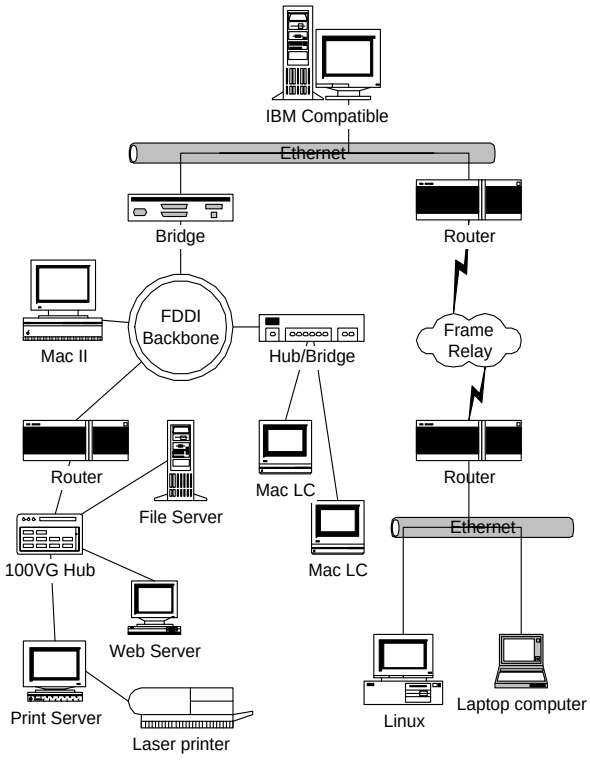

Figure 1 - An heterogeneous network configuration.

One of the problems associated with the current management frameworks, namely the SNMP (the most popular near data networks) is the absence of a robust and secure mechanism to

perform remote configuration. The manager has to move close to the dysfunctional system, check the problem and correct it. This solution is not effective and it is even less if the network is spread along a wide geographical area (in an extreme situation, another city or even another country).

Moreover, the diagnostic and corrective procedures can be very distinct even if we are dealing with the same type of problem but occurring in different systems and at a different time.

The current developing efforts are turning heterogeneous local area networks into a more uniform environment. Also the construction of distributed applications is getting easier and faster:

- Evolving application programming environments, as the Java language, define a common layer over heterogeneous physical platforms.
- The evolving of distributed programming environment (DPE) such as CORBA combined with well known proprietary protocols, such as Sun's RMI (Remote Method Invocation) [6] and Microsoft's DCOM (Distributed Common Object Model).

The level of automation in Network Management Systems is low or even null. It is resumed to e-mail messages or a pager warning sent to the manager when some working parameter gets out of predetermined bounds or some abnormal situations has occurred. The system, by itself, does not take corrective actions, not even in simple situations such as the reconfiguration of bad subnet masks or warning the user when disk space is insufficient.

The complexity of applications and systems is continuously growing. In a short period of time, the manager will be incapable of doing all by himself. The first step in automatic management could be done by automatically performing scheduled tasks through time, easing the burden caused by repetitive tasks. The next step could be the development of intelligent network management systems, based on an abstraction layer over extensive management data.

III. Management Information

A network management system contains several agents which are accessed by a management station. The management information is viewed as a collection of managed objects, organised in a Management Information Base (MIB) [7]. Every SNMP agent in the network implements one or more MIBs. The MIB is a virtual information store, defining managed objects. Objects in the MIB are defined using an abstract language - Abstract Syntax Notation One (ASN.1) [9]. In particular, each object has a name, a syntax, and an encoding [10]. Each MIB specifies variables needed for monitoring and control of various components in the network. A MIB potentially represents a large collection of objects. So, in a relatively large network, the provided management information can be enormous.

Although, this information is not redundant, its usefulness will vary for one management application to another. The bottom-up approach of SNMP management framework have provided thousands of information that may not have potential use. Also, the form in which object are available is not always adequate to management operations. For instance, the number of input datagrams discarded due to errors in their IP headers (1.3.6.1.2.1.4.4 - `ipInHdrErrors`, in MIB-II [8]) is not as helpful as its derivative with respect to time: the error rate.

A. Meta-variables

The meta-variable concept is not new [4]. A meta-variable is defined as a function over MIB variables allowing aggregation and data processing.

A meta-variable processor can be defined as an intermediary device in the network whose role is to compute predefined expressions and export its result to an NMS. Strategically placed processors can implement useful interpretations of management information. It is also possible to extend the concept so that a meta-variable can represent any number of agents, for example the mean of all the `ifInErrors` (1.3.6.1.2.1.2.2.1.14 - MIB-II) in a network segment.

The meta-variable processor is a process independent of the network management system. It can be seen as a proxy agent accessed in several different ways.

B. A Distributed Database

The distributed nature of a management system can be viewed as a distributed data base (Figure 2).

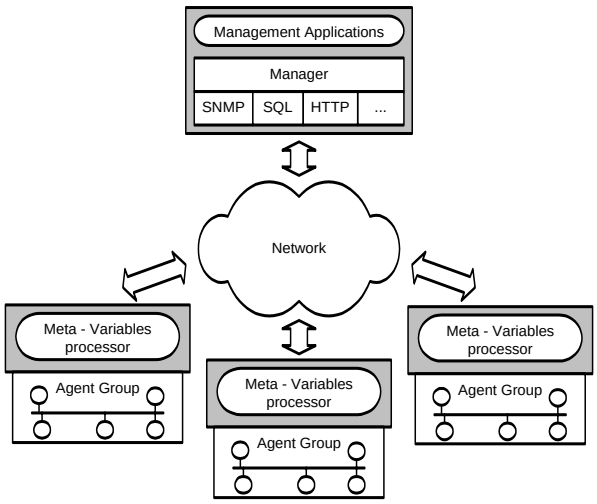

Figure 2 - Management system as a distributed database.

The information in the database comes from all the meta-variables and agents in the network and can be accessed in a number of ways with little changes to the presented architecture.

The meta-variable processor allows different types of access:

- as a proxy agent, accessible by SNMP;
- as a HTTP client, associated with a Web server and CGI.
- as a remote call, provided by mechanisms such as CORBA, RMI or DCOM.
- as SQL interface, considering the overall management data as a single virtual database [4].

IV. Network Management System

We have developed a Network Management System (PRONET) with by the following main goals:

- Robust - since the system is a general controller for network anomalies it will be very unpleasant if itself can suffer from any “disease”.
- Modular - the successive technical advances, specially in this field, may turn application inappropriate in a short time period. By adding or replacing some modules, it is possible to extend the application life.
- Distributed - remote information processing helps reduce the management traffic on the network, increases systems performance and creates a scenario less sensitive to catastrophic errors.
- Standards conformance - follow the standard is extremely important if we want to ensure compatibility in an heterogeneous environments.

PRONET is based on the Java language. The main reason for the choice is its distributed nature: it runs in any Java enabled browser and provides an easy to use distributed programming environment (RMI). In addition, there are very complete and powerful graphical toolkits, such as [12], that reduces the development time.

A. Global Architecture

PRONET is divided in two larger blocks: the NMS Applet and NMS Server. The primary role of the NMS Applet is to provide user interface and application-level functionality. As an applet, it can be executed in any platform capable of running a Java enabled browser.

For security reasons, a Java applet cannot access the disk in the local host or to perform connections others than to the server. Due to this limitations a Java application, running on the server and acting as a trader, receive and forward messages between the NMS applet and meta-variable processors. The HTTP server provides some bootstrap capabilities, when loading the applet. After the applet takes control, it uses RMI to communicate with the server.

B. Network Management System

Every network management system should have tools such as logical topology discovery, a network browser feature, MIB browser and some general SNMP utilities. In the PRONET, these functions are performed by the Network Browser/Management tools [13] (Figure 3).

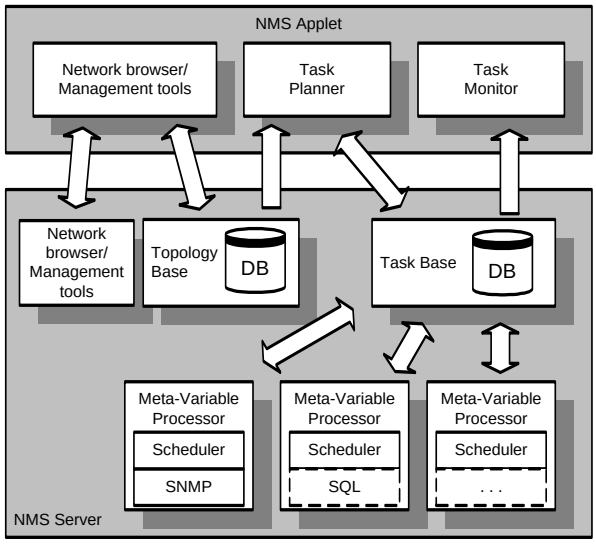

Figure 3 - NMS in detail.

The work window uses the Explorer paradigm to represent the network topology that is modelled through three main entities (Figure 4):

- Domain ( ) - representing an IP domain;
- System ( ) - an IP equipment with or without a SNMP agent;
- Group ( ) - associates several systems into one specific set (for instance, routers, printers, www_servers, etc.). The main idea is to group components with similar management requirements.

Some tools, as an Agent Browser, a Graphic Tracer and a Trap Log where also integrated. The Agent Browser is the trivial MIB browser that is common to almost management applications. The Graphic Tracer allows to specify several

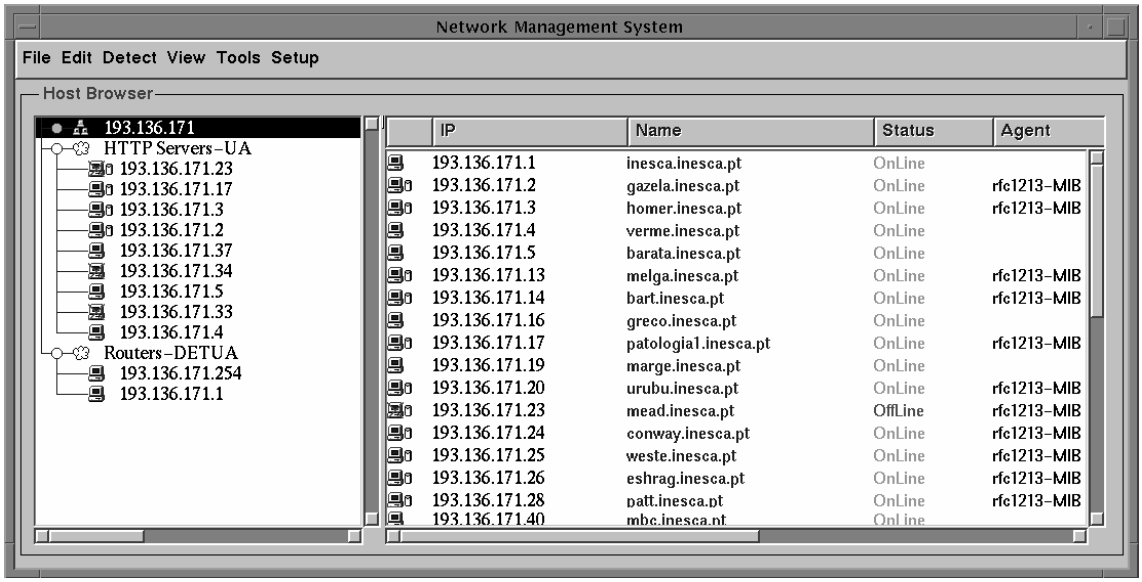

Figure 4 - Network browser/Management tools.

MIB attributes used to poll and draw the respective values. The Trap Log permits to register all traps generated from error situations in SNMP agents. All these features are performed in parallel using threads.

The topology information is stored in the NMS Server in a Topology Base. It gathers host information such as its network address, agent type, host status (`online`/`offline`) and group information, such as domain and logical grouping.

C. Data Abstraction

The NMS architecture include yet an information abstraction layer based on the meta-variable concept. The purpose is to provide a set of tools to simplify the elaboration of different and more useful views of management data. The definition of a meta-variable processor consists of two steps: a) the definition of an operation on the raw management data (Figure 5) and b) the scheduling information setting.

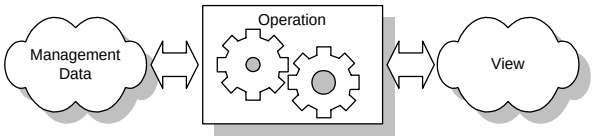

Figure 5 - Raw data manipulation.

The scheduling information combines three variables: `StartTime`, `Period` and `EndTime`. It defines the instant when the operation should be executed for the first time, the periodicity of the operation and when it should stop. The combination between operations and the scheduling establish a management Task.

These operations are executed in the Task Planner (TP) (Figure 6). It has a predefined set of functions that can be classified into three categories:

1. Simple algebrical functions: sum, minus, divide, multiply.
2. Complex algebrical functions: mean, derivative, etc.
3. General functions: threshold, mail, warn, graph, log, etc.

Every function implements the same template (in an OOP approach) (Figure 7). In Java, this means that they share the same `Interface`. This fact allows to wrap any number of functions so that it may be possible to build a complex operation based on the existing functions.

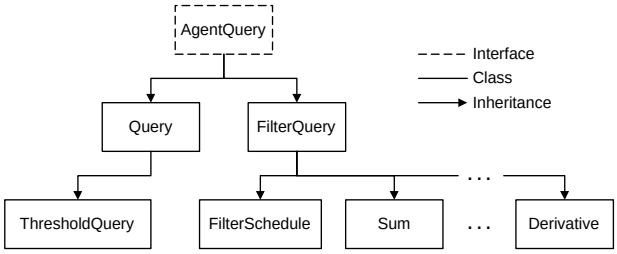

Figure 7 - Predefined functions inheritance tree.

Each operation is built by attaching objects (operation components):

```
...
agentObject=new Query(...);
schedule=new Schedule(...);
AgentQuery query=new FilterSchedule(schedule,
  new ThresholdQuery( boundListener,
  lowerLimit, upperLimit, agentObject));
...
```

The `query` object represents a periodic (`FilterSchedule`) query on an `agentObject` object. If the query value falls outside the integers `lowerLimit` or `upperLimit`, the object `boundListener` is notified.

This approach allows to associate any number of functions by

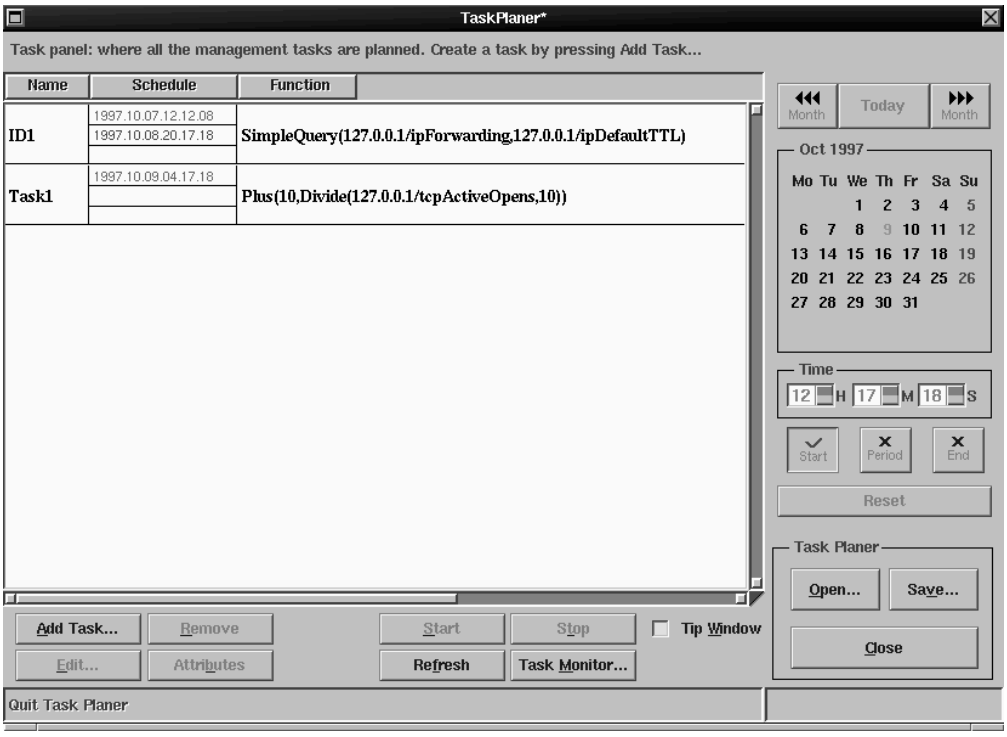

Figure 6 - Task Planner.

any order.

To the user, the creation of a management task is conducted by an expression wizard (Figure 8).

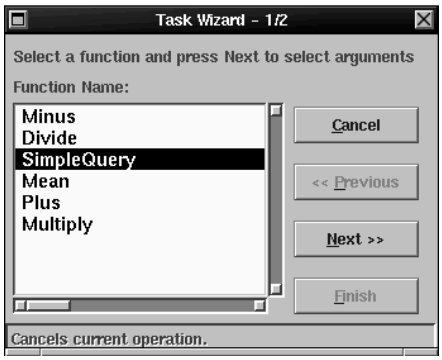

Figure 8 - Task Wizard (1/2).

It follows two steps:

1. Selection of the function;
2. Selection of the function arguments (Figure 9). If the selected argument is another function, it jumps to 1. recursively.

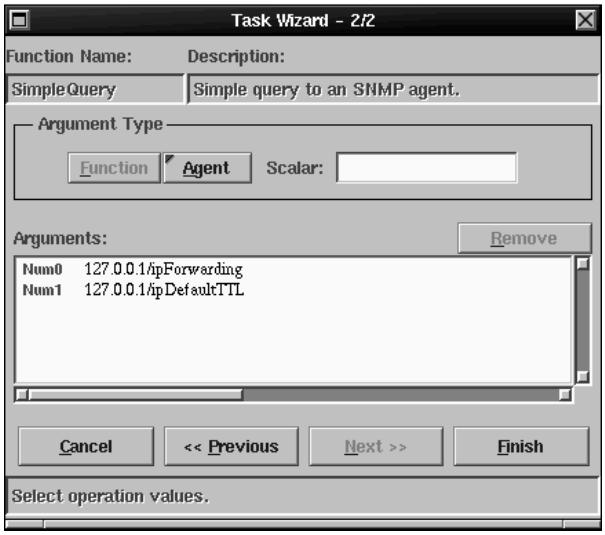

Figure 9 - Task Wizard (2/2).

After the operation definition, the scheduling information must be described. The TP has a calendar control that allows the definition of the `startTime`, `period` and `endTime`.

After the definition of the operation and the programming of the scheduling information, the server is updated with the new management task .

Similarly to the Topology information, each task is stored in the Task Base. Each object runs as a distinct thread. This means that it has its own resources within the global application. The thread approach eliminates the need for the application to maintain control on every meta-variable processor. In another words, each thread is responsible for its own scheduling control, information retrieval and data processing.

V. Proactive Procedures

The doctor/patient paradigm is the classical example of intelligent corrective measures taken on a faulty entity. The model is not exclusive to the health domain. It can be applied to the network management field.

The doctor, when examine a patient follows three steps:

1. Collect health data (symptoms): attend to patient complaints, listen to the organism vital signs;
2. Match symptoms to known health problems (diagnostic);
3. Generate the solution for the problem (heal).

The doctor generates the diagnostic based on a large health problem database, collected during the student years and as a result of the medicine practice. This is the bigger problem when transposing the scenario to network management field.

It is possible to transpose the previous situation to a network management scenario, attending to the data involved and the operations to be performed (Figure 10).

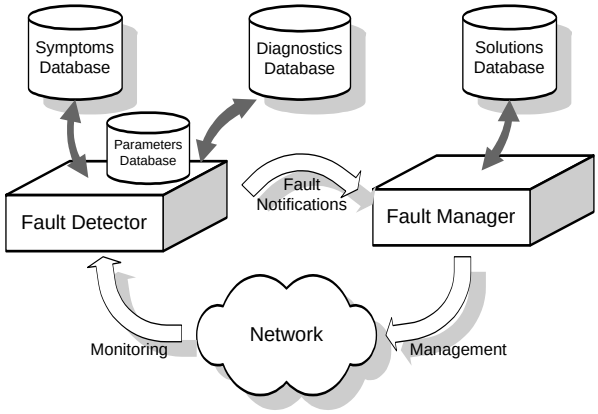

Figure 10 - Fault Management Architecture.

1. Collect management data (monitoring): listen to network components complaints (traps) and pool for management data. Each value is matched for sanity in the Parameters Database (Fault Detection). If the value is considered outside sanity bounds, a Symptom is generated.
2. The Symptoms from the previous point are correlated with the Diagnostics (Diagnostics Database). If the Symptom is recognised, a diagnostic is formulated: a Fault Notification is generated.
3. The Fault Notification is matched for an existing solution (Solutions Database). If it exists, the Fault Manager takes the corrective measures (Management).

The user intervention on such a system could be reduced to a minimum, leaving to the manager only unresolved situations.

VI. Conclusions

The management of a modern data networks deal with many problems. Network management systems does not provide sufficient automation mechanisms to help the user with some repetitive management tasks.

The SNMP framework approach have defined a very low level model, mainly concerned to management information

organisation and retrieval. Between this layer and the customers needs there is a large hole to fulfil.

This paper have presented a distributed Network Management System architecture based on data abstraction and on automatic procedures. It was also proposed the doctor/patient metaphor to detect and solve anomalies in a faulty network.

References

[1] Nat Brown, Charlie Kindel, *Distributed Component Object Model Protocol -- DCOM/1.0*, Internet Draft <draft-brown-dcom-v1-spec-00.txt>.

[2] OMG, *The Common Object Request Broker: Architecture and Specification, v.2.1*, August 1997.

[3] José Luís Oliveira, J. Arnaldo Martins, "A Management Architecture based on Network Topology Information", *Journal of Network and Systems Management*, Plenum, Vol. 2, No 4, pp. 401-414, December 1994.

[4] José Luís Oliveira, *Arquitectura para Desenvolvimento e Integração de Aplicações de Gestão*, PhD Thesis, Universidade de Aveiro, September 1995.

[5] José Luís Oliveira, J. Arnaldo Martins, "Scheduling and Processing Network Management Operations", *in Proc. of the 15th IASTED International Conference APPLIED INFORMATICS*, February 1997, Innsbruck, Austria, pp. 387-390.

[6] Sun Microsystems, Inc., "Remote Method Invocation Specification", *http://java.sun.com/products/jdk/rmi/*, 1996.

[7] M. Schoffstall, M. Fedor, J. Davin, J. Case, "A Simple Network Management Protocol (SNMP)", *Internet Request for Comments 1157*, October 1990.

[8] K. McCloghrie, M. Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", *Internet Request for Comments 1213*, March 1991.

[9] ISO/IEC 8824, Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One.

[10] K. McCloghrie, M. Rose, "Management Information Base for Network Management of TCP/IP-based internets", *Internet Request for Comments 1156*, May 1990.

[11] Chris Wellens, Karl Auerbach, "Towards Useful Management", *The Simple Times: The Bi-Monthly Newsletter of SNMP Technology, Comment and Events*, Volume 4, Nº3. ftp://www.simple-times.org/

[12] BISS GmbH, *The BISS Java framework*, http://www.biss-net.com/biss-awt.htm

[13] Gabriel Vieira, Orlando Sá Morais, *Bancada de Gestão de Redes*, University of Aveiro, September 1997.