Dynamic Deployment of Value Added Management Services for Actives Networks

Mauro Sergio P. Fonseca [1,2,*]
mauro.fonseca@lip6.fr

Nazim Agoulmine [4,3,1]
nazim.Agoulmine@lip6.fr

Université de Paris VI – Lab. LIP6, France [1]
Université d'Evry, LSC, France [3]

Pontifícia Universidade Católica do Paraná, Brasil [2]
Université de l'UQAM, Canada [4]

Abstract

The evolution of Internet during the last decade was impressive. However, as interconnected networks and customer's increases the management of the overall services became more and more complex. The traditional approach of IP network management SNMP (simple Network Management Protocol) and the corresponding MIB (Management Information Base) model even if they are widely deployed, they do not respond completely to the newly requirements. In fact, the size and the heterogeneous underlying technologies necessitate a more flexible, consistent, reliable, and scalable approach.
One of a possible potential solution is Active Networking that aims to make more intelligent the intermediate routers in the network and possibility to download code directly in the core of the network to influence its behavior. Thus, this paper aims to propose an active management framework as a flexible solution to IP networks and services management. The framework proposes management architecture as well as a set of mechanisms to replace SNMP and its SMI (system Management Information) static information model by a new active protocol and a Dynamic MIB (DMIB).
The DMIB improves the management of large complex networks by moving management decision points closer to the node being managed and dynamic deployment of new manager and managed objects.

Key words: "Dynamic", "Active Network", "Manager".

1 Introduction

The development of digital systems in telecommunications and the advances made in field programmable gate array (FGPA) has shifted the emphasis from the hardware of network equipment's to the software they run. Thus, it is clearly desirable to deploy in the future as open as possible telecommunication network infrastructures incorporating computational capabilities directly in the network nodes. As a matter of that, it is possible to develop new telecommunication services that are able to move some of their functionalities, that has been traditionally located in end systems, directly in the core network without affecting the hardware equipment's already deployed. The network is than considered as an active part and a manageable resource of the running services and not only a passive communication tube.
The work presented in this paper is related to the management of such networks as well as the active services running on the top of these active networks. The aim is to enhance existing management protocol to take advantage of the active network facilities in order to propose a

* Scholarship from CAPES / BRAZIL

flexible and adaptable management solution to service and network monitoring and control. The proposed architecture is called **ANSMA : Active Network and Service Management Architecture**. This approach permit to render completely dynamic the way network equipment's and services can be managed according to administrator policy or the running services themselves.

2 Background concepts

The requirements in terms of management of IP based network can vary from one provider to another and from one deployed service to another. In the actual context without regularization, ISPs (Internet Service Providers) are in a high competition context and are willing to control in more precise manner their resources. One can be interested only by the performance information while the usage resources in the network can interest another. ISP wants to deploy in a dynamic manner different management strategies according to their business objectives. For instance, an ISP can be interested in controlling the duration of multimedia services at one time and controlling the quantity of exchanged information at another time.
Traditional approach of management makes this task very complex, this is important to propose a new approach to permit this kind of dynamic in a flexible manner.

2.1 IP Network Management

The IP network management is mainly base on the Simple Network Management Protocol. The idea behind SNMP is relatively straightforward and easy to understand. A single manager may control many agents. As shown in figure 1, the SNMP is built upon the connectionless UDP Transport Protocol [1]. The manager uses SNMP request to retrieve information from the agent in an asynchronous manner. The consequence of using UDP as a transport protocol is that managers should perform some kind of polling to detect whether agents are still operational. Agents can send traps to notify some events, however those can be lost in the network and the manager can completely ignore the event. The SNMP protocol does not itself perform retransmission [4]. The behavior of the agent is completely static, meaning that it is not possible to change or enhance its capabilities during time. Its capabilities are reflected by the set of information represented by managed object into a Management Information Base, which can be manipulated by manager.

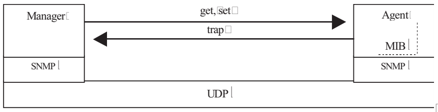


Figure 1 : SNMP agent-manager model

2.2 Active Network Technology

Active networking emerged from discussions within the Defense Advanced Research Projects Agency (DARPA) research community. Enabling the programmability of networks was expected to allow a flexible network architecture, which would ease the integration of new services and would accelerate network infrastructure innovation. Two approaches are being discussed. The first one, called "programmable networks", is to open up network control to applications by means of standardized interfaces that permit to control the service logic of programmable switches or routers. The second approach is based on the transfer of executable code by means of "active" packets or capsules. This latter approach is more dynamic approach and enables a more efficient provisioning of network services, which can be dynamically customized for particular types of applications, users or individual packets. However, it introduces many problems of security, reliability, resource allocation and availability.

2.3 Motivation for using AN management

The main advantage introduced by AN is its capabilities to allow end user services to define their own mechanisms to process packets in intermediate nodes, thus it is possible for them to dynamically download new functionalities directly inside the routers depending on their activities. Management of such services needs to take into account this dynamic behavior that cannot be fixed initially in a static manner as with SNMP where the node MIB is initially identified [3,5,6].

Thus an Active Network and Service Management approach needs to define a dynamic approach to management information specification and management functions deployment.

Consequently, the target system should provide:

- The possibility to dynamically extend management information in the management information base.
- A protocol to download this information directly in the target active node.
- The possibility to enhance management function in active node by recombination with existing management services.

In our approach, we use the natural capabilities of the underlying active network framework to adapt the management capabilities of active nodes during service execution.

2.4 Concept of Dynamic Management Information Base

The need to dynamically extend the management objects within a node has given rise to a variety of management instruments. The approach to dynamically define management facilities was already introduced by the university of Columbia in the frame of Extensible Agent [13]. However, it was necessary to extend some how the SNMP protocol to encapsulate the new functionalities. Since, wide varieties of approach have been proposed without a common agreement, applications have to support several Application Programming Interfaces (APIs) in order to be operational. In Active Network, specifying a standard API for dynamically extend the management objects and information provides the technical foundation required to solve this problem.

2.5 Role of an Active Management Protocol

As the node number and complexity increase, management centers become points of implosion and inundated with large amounts of redundant information. It is essential that networks management interactions employ techniques that require less communication and permit more effective action on the managed node itself.

Thus an active management protocol should be defined in order to support the interaction between the management center and the active nodes. The goal of this protocol is to support:

- Node dynamic enhancement of management information and functionalities.
- Management application deployment in order to deploy efficient solutions.

2.6 *Advantages of Management service composition*

Because of the dynamic aspect of applications, ideally, active service will deploy active code that realize the functionalities to process packets in intermediate nodes as well as specific management code of this code. The management centers can take advantage of these capabilities in order to deploy high level management services that are able to interact with specific management code embedded in active nodes depending on a predefined high level component interaction design. This will facilitate the specification and the deployment of new management applications based on existing previously installed management capabilities on active nodes. However, this problem is not addressed in this paper but is an ongoing research hope.

3 Proposed ANSM Architecture

The proposed Active Network and Service Management Architectures, as show in figure 2, consists of:

- Dynamic Information (DInfo) interface what is an API to access managed information or service objects. This API is composed of five primitives: getAbout(), getVersion(), get(), set() and function().
- Dynamic Management Information Base (DMIB) interface what is an API to access DInfo's objects. This API is composed of the following primitives: getAbout(), getVersion(), get(), set() and function(), getObject(), nextObject(), restart(), putObject(). To ensure the unique identification of each DInfo, the DMIB uses the same principles as the SMI naming tree [2,8]. Where the leaves of this tree represent the DInfo's objects.

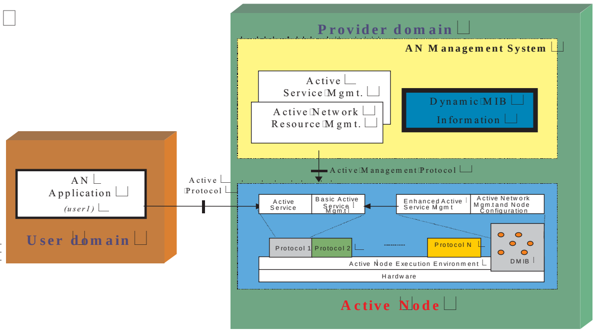


Figure 2 : Active Network and Service Management Architecture

The AN Management System: This component is the central point of management of the provider domain. All the management policies are defined within this system. It comprises a set of tools that enable network manager to define the basic management functionalities to be deployed in the network. These functionalities concern the network level as well as the service level. Each functionality can be developed based on a service composition principle which means that the manager can reuse existing management components already deployed in the managed active node.

Managed Active Node: managed active nodes are the active nodes that are controlled through the active management system. They contain a DMIB that offers an abstract view of the managed resources of the active node. The managed active node, runs an active node execution environment that support active protocols.

3.1 The Dynamic Management Information Base

The DMIB framework describes how the set of managed objects and information supported by a particular active node may be changed or added dynamically. It defines objects called DMIB, Dynamic Information (DInfo) and the elements of procedure by which the Dynamic MIB objects behave.

The Dynamic Management Information Base (DMIB) objective is to describe and to standardize the way to the set of managed objects, information and manager objects supported by a particular active node may be changed or added dynamically by a remote management system.

The idea behind DMIB is to improve the management of large networks by shifting the management decision points from the centralized management center directly to the active node being managed. A single manager may control many Active Nodes directly or indirectly. As shown in figure 3, the DMIB is an Active Node built in object. The advantage to use built in object approach of DMIB implementation is to provide a complete open solution. This means that any external manager can dynamically deploy new DInfo using active capsules.

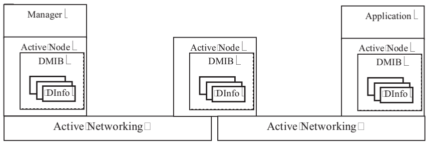


Figure 3 : The Dynamic Management Information Base framework

The DMIB mainly consists of two components:

- The DInfo that represents object encapsulating information as well as methods to control these information. Methods that give access to local DInfo information as well as information from others DInfo for a management purpose.
- The DMIB allows changes to be performed on DInfo from remote managers. The DMIB naming tree is used as a mechanism for information identification in the network, as described in Figure 4. The tree leaves represent the DInfo objects. This structure permits to dynamically enhance the tree with new DMIB or DInfo object at any time.

A DMIB can be composed by 0,1,2 ...N DMIB or DInfo objects.

A DInfo cannot contain any DMIB or DInfo objects.

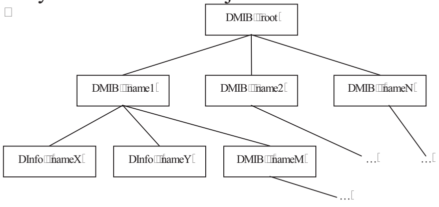


Figure 4 : DMIB - structure of naming tree

Managed objects can be accessed through a virtual information repository of DInfo through five primitives. For example, we can employ DMIB to implement a dissemination algorithm

to populate different DMIB in different active nodes using DInfo. The objective is to shift a management decision points from the management center directly into active node.
Normally when a dissemination algorithm approach is used in active node, the main result will be the decrease the communication traffic between the manager and the active node, as presented in Figure 5, only treated events are reported to the manager; through this technique is feasible to replace SNMP Traps [7] and other features with more scalability.

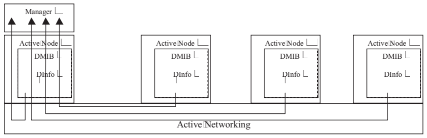


Figure 5 : DMIB - Dynamic Information send Information to Manager (asymmetric)

Hence as shown in Figure 6, it is possible to configure the DMIB to makes it work as the classical SNMP, model interactive. This allows to quickly adaptation of SNMP manager to a DMIB manager.

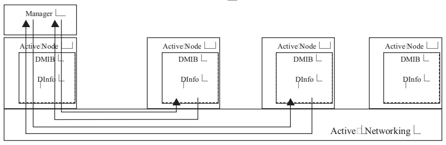


Figure 6 : DMIB - get Dynamic Information like SNMP

It can also be used to a more efficient approach to collect information from different active nodes. This approach requires simultaneous asynchronous activities using a get primitive of DInfo, as shown in Figure 7.

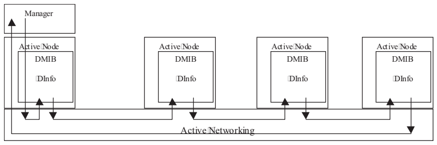


Figure 7 : DMIB - get multi Dynamic Information

These schemes show the flexibility to implement different management strategies and the dynamic management information change using DMIB. It specifies a standard API for dynamically extend the management objects and information, providing the technical foundation for negotiation of new products in a Unified Manager Framework.

4 Dynamic Information (DInfo)

DInfo objective is to replace the managed object of the MIB and provide de infrastructure necessary to deploy dynamically new services and managed object. It is a set of information and related method that enable accessing and controlling management information. DInfo uses an "object oriented" approach to represent and exchange management information.

Many management rules usually used in management center can be downloaded directly in the target management node. Management center rules can be embodied in DInfo that is downloaded in remote managed nodes.
Thus its is possible to automatically identify and correct problems without requiring help from the manager center. DInfo allows management customer to employ techniques that require less communication and thus permitting more effective self action on managed nodes.
Dynamic Information is formed by information (About, Version and other information) and functions. Function is the procedure performed by DMIB parent. This permits self manipulation of information or manipulation of other DInfo's.
A function in DInfo can manipulate DInfo(s) to actualize information or to apply rules to manager or other actions. As DInfo is an object, it is possible to implement the API and have many other information or functions aggregated to it to support different objectives (rules, state, etc.). The proposed standard (API) for DInfo provides the technical foundation required to solve problem of specific MIBs development too.

The DInfo API is:

getAbout()	Return a String with About information of the current DInfo object (purpose, functions, information and correlated).
getVersion()	Return a String with version information of the current DInfo object.
get()	Return an object. This object should be specified in About information.
set()	Set an object. This object and yours range valor should be specified in About information.
function()	This method is responsible to implement the specific DInfo treatment (to manager, to actualize or other functions). It is used by DMIB parent to scheduler this object.

Figure 8 : Table with DInfo API

5 Dynamic Management Information Base (DMIB)

Various IETF working groups are continuously defining new MIB modules that extend the Internet standard MIB. It is also common for enterprises or individuals to create or extend enterprise specific or experimental MIBs. As a result, managed devices are frequently complex collections of manageable components that have been independently installed on a managed node. Each component provides instrument for the managed node and for the managed objects defined in the MIB module(s) it implements. The DMIB approach offers a solution to this problem in Active Network.
DMIB is a built in component of node that permits to add new DMIB and DInfo objects dynamically. There is a DMIB root that is responsible by scheduling and to perform specific functions. The nesting structure of DMIB allows the creation of independent domain name inside each DMIB. As in each DMIB there is a self function, it is possible for customize the scheduling to objects (DMIB's and DInfo's) inside.
With the DMIB mechanism, it is possible increase the DMIB with new DMIB in order to group new DInfo or only new DInfo at any time without reinitialize the active node. As DInfo has information and function, it is possible to insert new DInfo with new function,. Now is possible to insert or extend managed node dynamically with new management rules in order to move or up date the node with new manager functionality.

A DMIB consists of:

- A single or multiple DMIB structured tree, which DInfo, but with no direct access to management information.
- A single or multiple DInfo, which have direct access to management information.

The DMIB performs the following functions:

- Accepts request from management Capsules to access DMIB and DInfo objects.
- Accepts request from management Capsules to add new DMIB objects in the tree.
- Accepts request from management Capsules to create new DInfo object in DMIB.
- Schedule all DInfo functions when applicable (if DInfo is different from NULL).

The DMIB API is:

getAbout() *	Return a String with About information of the current DInfo object (purpose, scheduler functions, information and correlated).
getVersion() *	Return a String with version information of the current DMIB object.
get() *	Return an object with the next DInfo object of the current DMIB object.
set() *	Without function, it is to be DInfo API agreement only.
function() *	This method is scheduled by DMIB parent, and is the guarantee that it is possible to customize a new scheduler to DInfo's in this DMIB.
getObject (name)	Return the DInfo object with the matched name.
nextObject ()	Return the next DInfo object in DMIB.
restart()	Initialized to first DInfo object in DMIB to next use of nextObject ().
putObject ()	Put new DInfo or DMIB object in DMIB.

* DInfo API agreement.

Figure 9 : Table with DMIB API

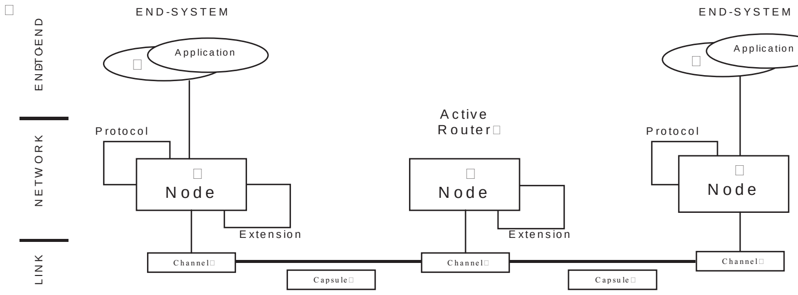
6 Prototype

To implement the ANSMA prototype, we have used the ANTS Toolkit. The motivation is that ANTS is a full Active Network platform, straightforward and useful. The decision has been done over Active Networking Systems Evaluation [9].

6.1 ANTS Toolkit

The Active Node Transport System [10,11] is based on mobile code, loading on request and caches techniques. It permits new protocols to be deployed dynamically in network nodes, without synchronization or interaction between the physical nodes and running protocols. The architecture can be described using the three following key concepts: (1) traditional IP packets are replaced by capsules that customize network services; (2) classical routers are substituted by Active nodes that proceed incoming capsules and maintain soft store; (3) a code distribution mechanism is used to dynamically and automatically deploy routines to the nodes requiring them.

The ANTS defines the concept of active node connected by *Channels* to form a network. *Capsules* are injected into the network by *Applications* and forwarded to the destination with customized routines. In the prototype implementation of ANTS [12], these concepts are



mapped to Java classes as described in Figure10.

Figure 10 : Key classes of the ANTS toolkit and their relationships.

6.2 DMIB implementation

To implement ANSMA within ANTS toolkit, we have specified a DInfo interface as described below:

```
public interface DInfo {
   public void function();
   public String getAbout();
   public String getVersion();
   public Object get();
   public void set(Object ob);
}
```

The DMIB class implements the interface *DInfo* and extends a *Thread* class, to be a new thread in the class Node to improve the scheduler objects DInfo complied, through the method *function ()*. The DMIB class uses the *Hashtable* class to enhance the control of DInfo and DMIB deposit object. The DMIB class implements also the others DInfo API methods, as shown and described in figure 9, as discussed before.

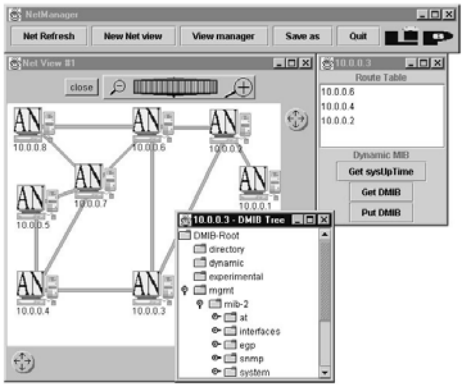


Figure 11 : ActiveNetManager an ANSMA Implementation.

Then, we developed a manager application to use the DMIB and the DInfo classes in ANSMA context. This manager is called ActiveNetManager, as showed in Figure 11. It has the following features: *net self discovery*, *net browser* (to interact with node), *DMIB browser* (like SMTP browser) and *tools* (to send new DInfo and DMIB to active node).
A set of active protocols was developed to make functional the ActiveNetManager. One important protocol is responsible for sending and installing new classes of DInfo or DMIB into the active node, as shown in Figure 12.

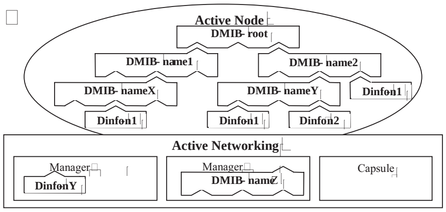


Figure 12 : DMIB - structure tree.

To accomplish this, the *sendDMIB()* class is the inherit from *DataCapsule* class of ANTS. This latter that can transport a data buffer until a destination node. In this case, the data buffer contains a customized class DMIB or Dinfo. When *sendDMIB* capsule arrives in an active node, it executes the method evaluate from the API capsule.
If the node address is different from the destination address, the capsule is forwarded to the destination Active Node, otherwise, it tries to get the DMIB root object from the local node, by the *getDMIB()* method to get the object DMIB *dynamic*, that means a branch of DMIB tree, as shown in Figure 11; and to get the object DMIB *LIP6* in DMIB object *dynamic* as shown in Figure 13. If there were not the DMIB object *LIP6*, it will be created and put in DMIB object *dynamic*.
We can verify in the Figure 11 the DMIB *LIP6* object has not been installed. However in the figure 13, we can notice the installation of a DMIB *LIP6* object and a DInfo *dteste* object, which will be executed and the DInfo object code implementation shown later.

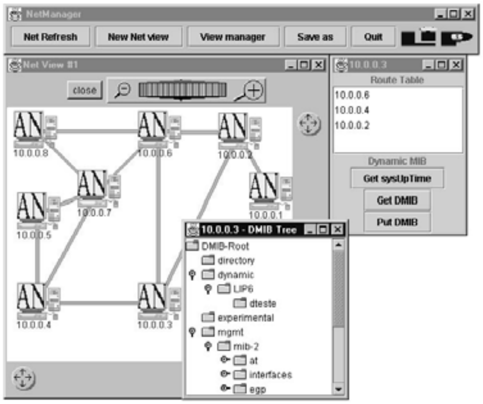


Figure 13 : DMIB Tree with dteste DInfo object.

Then, the method gets a customized class DMIB or Dinfo into self data buffer and writes it on the local node. After this, it tries to instantiate this customized object and fixes it into the DMIB *LIP6* object. After this, the new customized DMIB or Dinfo will be scheduled and executed. The Figure 13 shows the DMIB tree after the installation of the *dteste* DInfo object.

The method evaluate code to install the DInfo is presented.

```
public boolean evaluate(Node n) {
 if (n.getAddress() == getDst()) { // if node is destination node
  try{
   DMIB dmib = n.getDMIB(); // get local DMIB root
   DMIB dynamic = (DMIB) dmib.getObject("dynamic"); // get "dynamic" DMIB in DMIB root
   DMIB lip6 = (DMIB)dynamic.getObject("LIP6"); // get "LIP6" DMIB in dynamic DMIB
   if (lip6 == null){ // if there isn't LIP6 DMIB
    lip6 = new DMIB(new String("Test LIP6"), new String("0.1")); // create a new LIP6 DMIB
    dynamic.putObject("LIP6",lip6); // put new LIP6 DMIB in "dynamic" DMIB
   }
  ByteArray buf = getData(); // create a buffer
  FileOutputStream file = newFileOutputStream(path); // create a file to save new DInfo code
  nt = buf.length();
  file.write(buf.toBytes()); // save new DInfo code
  file.close();
  try{
   Class DInfoClass = Class.forName(className); // read new DInfo class
   if (DInfoClass != null) // if DInfo class OK
    lip6.putObject(dirName, (DInfo)DInfoClass.newInstance()); // put new DInfo class in "LIP6" DMIB
   } catch (Exception e) {n.log("ERRO"+e);}
  } catch (Exception e) {n.log("ERRO"+e);}
 } else {return n.routeForNode(this, getDst());} // if no is different of destination then go to destination
return false;
}
```

The DTest DInfo class code

```
package ants.mib.dynamic;
import ants.*;
import java.io.*;
import java.util.*;
public class DTest implements DInfo { // A very simple example
  private String about = new String("Dynamic Test"); // make an about explanation
  int cont = 0;
  public void function(){ // the function get and print "About" and "Version" information
   if (cont > 20){System.out.print(getAbout()+getVersion());cont = 0;}
   cont++;}
  public String getAbout(){return about;} // return the about
  public String getVersion(){return new String("0.01");} // return the version
  public Object get(){return (Object) about;} // return the about in this example
  public void set(Object ob){about = (String) ob;} // set the about in this example
}
```

7 Conclusion

In this paper, we have proposed an active management framework as a flexible approach to networks and services management. Dynamic Management Information Base is a concept that extends and replaces the traditional SNMP model through new features of flexibility and scalability. The DMIB was developed to allow manager applications to easily achieve common management goals and scalability. This solution belongs to the Active Networking ideas. The additional traffic introduced by capsule in active networks is the main problem addressed. However we think this overload will be compensated by the processing and traffic load saved by the deployment of management functions directly in Active Nodes.
At the moment, this works is progressing towards the implementation over ANTS in order to test different scenarios showing the flexibility, consistency, reliability, scalability, applicability, and consistency of this approach regarding traditional approaches.

8 References

[1] Postel, J.B., "User Datagram Protocol", RFC 768, August 1980.

[2] Rose M. T.; McCloghrie K., "Structure and identification of Management Information for TCP/IP based internets", RFC 1155, May 1990.

[3] McCloghrie K.; Rose M.T., "Management Informatio Base for network management of TCP/IP based internets", RFC 1156, May 1990.

[4] Case J. D., Fedor M., Schoffstall M.L., Davin C., "Simple Network Management Protocol (SNMP)", RFC 1157, May 1990.

[5] McCloghrie K., "Concise MIB Definitions", RFC 1212, March 1991.

[6] McCloghrie K.; Rose M.T., "Management Informatio Base for network management of TCP/IP based internets: MIB II", RFC 1213, March 1991.

[7] Rose M. T., "A Convention for Defining Traps for use with the SNMP", RFC 1215, March 1991.

[8] K. McCloghrie; D. Perkins; J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)", RFC 2578, April 1999.

[9] N. Achir; M. S. P. Fonseca; Y. M. G. Doudane; N. Agoulmine; A. Mehaoua, "Active Networking Systems Evaluation : A Practical Experience", MoMuC'2000, Tokyo, Octobre 2000.

[10] D. Wetherall et al., "ANTS: A Toolkit for Building and Dynamically deploying Network Protocols". In IEEE OPENARCH'98, San Francisco, April 1998.

[11] D. Wetherall, "Service Introduction in an Active Network", Ph.D. Thesis, MIT/LCS/TR 773, February 1999.

[12] D. Wetherall, "Developing Network Protocols with the ANTS Toolkit", Design Review, Août 1997.

[13] M. Daniele; B. Wijnen; M. Ellison; D. Francisco, "Agent Extensibility (AgentX) Protocol", RFC 2741, January 2000.