Hierarchical Management Protocol for Large Networks

Marcial Porto Fernandez [1]
E-mail: marcial@gta.ufrj.br

Gardel Moreira Delfino[1,2]
E-mail: gardel@gta.ufrj.br

Aloysio de Castro P. Pedroza [1,2]
E-mail: aloysio@gta.ufrj.br

[1]Grupo de Teleinformática e Automação (GTA)
Universidade Federal do Rio de Janeiro (UFRJ)
COPPE/PEE - Programa de Engenharia Elétrica
Caixa Postal 68504 - 21945-970 - Rio de Janeiro - RJ - Brazil
Tel: +55-21-260-5010 Fax: +55-21-290-6626
http://www.gta.ufrj.br
[2] Departamento de Eletrônica/ EE-UFRJ

Presented at LANOMS'99 - Latin American Network Operations and Management Symposium
Rio de Janeiro - Brazil - December 03 - 05, 1999

ABSTRACT

A computer network management system presents one manager and several agents. The management of large networks demands much computer resources and management system capacity, impossible to achieve with just one manager. The TMN standard, that intends to standardize management systems for telecommunication networks offering those resources, already foresees the use of several managers organized in a hierarchical architecture. This work presents a study of the problems in managing a large network and proposes a protocol for a Hierarchical Management System (HMS). Although we use a example of telecommunication network, the HMS system is suitable for Internet. The HMS protocol is specified using the formal description language Estelle and its functionality is verified using a test model.

Keyword: Network Management, Protocol, Distributed System, Formal Description

1. Introduction

Most networks management systems use the same philosophy: a main element should collect information and start commands to control the system, while other elements acquire this information and execute the orders. The main element is called *manager* and the elements that collect information are called *agents*.

This architecture is simple and assists most of the actual systems, but it begins to present problems when there is an increase in the amount of managed elements. This number of agents increases the processing needs in the manager, as well as the traffic in its communication channels. These are typical situations of telecommunication network management systems, in which the number of managed devices can reach several millions.

The TMN (Telecommunications Management Network) standards [1], published in 1992, proposed a system specification that allowed an integrated management of telecommunications network. ITU (International Telecommunication Union) included hierarchical management in the TMN standards, recognizing the need of realizing a system with several managers organized in a hierarchical structure, but they didn't define how this management would be implemented.

This work proposes a protocol that implements an open hierarchical management system, in agreement with the objectives of TMN standards. The architecture of hierarchical management is adapted to control a large telecommunication network; it limits the management to small groups, but allows the synchronization of management actions in the whole system. The TMN architecture is used to concentrate all the management information and execute the control of managed elements.

Section 2 presents problems in managing a large telecommunication network and some techniques of hierarchical management and management concepts used in our work. Section 3 presents the Hierarchical Management System (HMS) specification. Section 4 shows HMS implementation, the methodology used in project and the simulation prototype and tests. Finally, section 5 presents the conclusion of this work and propose some future enhancement.

2. Management Systems

This section presents the problems in managing a large telecommunication network. We will also present some techniques of hierarchical management in the Internet and distributed management concepts used in our work.

2.1 Problems to Manage a Large Network

The main problem of a large network is that it can not be managed by only one manager. Even with the increase of the new equipment capacity, we will always have a limit, sometimes due to processors capacity, sometimes due to communication channels speed. The system should be flexible and scalable, able to increase its capacity according to the needs.

The solution would be to implement a distributed system, with several managers, in a way that each one could manage a group of agents. But it is important to take into account that the existence of several managers could cause an authority problem. Then, each one should know which one can control and which one has to give information, so that the system is coherent and there is no conflict of instructions set by different managers. The system cannot link an agent to a manager in a rigid form, because in the case of a manager failure, another one should assume the control.

A hierarchical system should also respect the responsibility and authority of the companies that operate the telecommunication plant. As generally there are several companies operating the telecommunications in an area, it is necessary to respect the authority on each company's operation area.

2.2 TECHNIQUES TO MANAGE A LARGE NETWORK

2.2.1 MANAGER-TO-MANAGER MIB

The first technique of a system with more than one manager was implemented in Internet and is known as Manager-to-Manager MIB, specified by CASE et. al.[2]. A secondary manager joins information from agents and transmits management data, as alarm, events and notification, to the main manager. Moreover, this main element execute a database synchronization and avoids two managers to write different information in the same agent. Manager-to-Manager MIB is implemented in SNMP version 2.

The main objective of this technique is to allow remote managers, connected with slow communication links, to managing without degrading the efficiency of this channel. The Manager-to-Manager MIB is not adapted for managing large networks because it doesn't reduce the processing of information in the main manager, it just decreases the frequency of data transfer between secondary and main manager. No more complex process to make decision is provided by this system [3].

2.2.2 HIERARCHICAL MANAGEMENT - SUB MANAGER

Sub Manager is a technique proposed by SIEGL and TRAUSMUTH [4]. It extends the Manager-to-Manager MIB concept, increasing the functionality and allowing more hierarchical levels. The basic idea is create a device called Sub-Manager, between the manager and the agents. That device is composed by a control entity and three tables: SubMgrEntry, that stores agents data; SubMgrOps, that stores a list of procedures (script); and SubMgrValue, that stores the results of the operation of SubMgrEntry by the procedures of SubMgrOps.

2.2.3 HIERARCHICAL MANAGEMENT - META-MIB

A technique to represent a physical configuration in the Internet, considering hierarchical and distributed management concepts, is called Meta-MIB. This technique is used in the WILMA Project and it is shown in SCHALLER [5]. The idea is to define three new MIBs: Components Table, specifying the type of equipment in each IP address; Connections Table, specifying the connections of all equipments; and Agent Table, specifying the state of agents information base.

2.2.4 DOMAIN MANAGEMENT

The Domain Management is a proposal of SLOMAN et al. [6]. It defines the relationships among the several elements of the system. In a large management system, it is difficult to control and visualize all the elements of the network. Any manager, computer system or human operator, should join several objects into only one group. The concepts of managed object, composite object and managed object are shown below.

A managed object is defined as an entity that has management resources. The main element is the management core, that is composed by a control software and MIB table. This core interacts with the environment through the following interfaces: a communication interface with the upper level device, a communication interface with the lower level device and an interface with the local resource.

A composite object is defined as a group of managed objects with similar characteristics. This group is considered as only one managed object that contains all the internal objects. A manager object groups all the information collected from managed objects and represents that whole group of information as one object only.

2.2.5 Policy Management

The Policy Management is also a proposal of SLOMAN et al. [6]. The management of distributed systems involves the functions of monitoring the activities, taking management decisions and executing control actions to modify the behavior of the system. Due to its own philosophy, the manager should have autonomy to decide the actions that he should take, without needing to receive an explicit order from the upper level manager. These decisions, however, inside a distributed system, with several objects executing management functions, need a policy that guides its behavior.

The specification of a policy by a human manager begins in the highest abstraction level and it will be refined at each level below. It should be noted that a policy doesn't necessarily specify what to do for a certain situation, but it defines the decision parameters. The policy is not simply a value that should be assisted. In a heterogeneous system, with different equipments from different manufacturers, this policy can take different values depending on the function or characteristics of each object.

Another policy management proposal was made by WIES [7]. The basic idea is similar, it just complements the previous proposal, classifying the policy according to more specific approaches.

2.2.6 Delegation Management

The concept of delegation management, was proposed by GOLDSZMIDT and YEMINI [8]. It specifies a feature to make the management system flexible, that is, a system adapted to the size of domain along the time. The idea is that a manager, when detecting that it is overloaded to manage a group of objects, can delegate the management of some objects to other manager. The author's approach is that part of manager software is given to other manager that executes this code.

Another proposal was made by MOFFETT and SLOMAN [9] and that was used in the DOMINO Project , that, together with control authority, also specifies delegation authority. The system can impose restrictions similar to human organizations, where a boss can delegate the leadership to a subordinate, but he cannot delegate it for another person without his boss's authorization, for example. HMS uses this approach.

2.3 A Proposal for a Management System for Large Networks

The TMN standard adopt the CMIP [10] as its management protocol, thus, our system will use it, but the system can be easily adapted to use SNMP. We have used the Meta-MIB approach, that gives the possibility to create a true hierarchical architecture. The most interesting Meta-MIB table for our system is the Agents Table, that allows the representation of several hierarchical levels in the management system. In our proposal, we have used a structure called Meta-MIB, that implements the function of the Agents Table.

Our system is a combination of this technique with the concepts of Distributed Systems, which provides flexibility. The Domain Management permits the control of the managers and agents position in the hierarchy. The Policy Management allows to organize the command authorization in the system, defining limits to all of the managers that have autonomy without interfering in the global control of the system. The Delegation Management offers resources for the system to make the load balance, allocating agents to another managers.

3. Specification of Hierarchical Management System (HMS)

In this section we will show a management system able to control a great number of devices, using a hierarchical architecture. The base of this architecture is the conception of devices that have two different behaviors: as a manager (in relation to the several agents under its control) and as an agent (in relation to the manager that controls it).

3.1 STRUCTURE OF THE HIERARCHICAL MANAGEMENT SYSTEM (HMS)

The Hierarchical Management System (HMS) is a module at the OSI Application layer, at the same level of the TMN module. The organization model of the proposed system is presented in Figure 1, that shows the disposition of several modules. The lowers layers are represented as Communication Layer. Above them, there is a layer that offers the CMIS (Common Management Information Service) service and use the CMIP protocol .

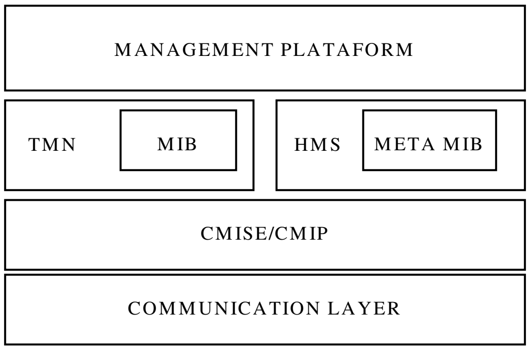

Figure 1 - Management System Architecture

The TMN module executes the functions of collecting all agents information and putting them in its Management Information Base (MIBs). This module implements the manager behavior and assists the management needs of a telecommunication network. The higher layer was called Management Platform and consists a graphical interface that offers the management information for the system operator.

The HMS module executes the functions of Domain Management, Policy Management and Delegation Management. It assumes agent behavior, supplying information and receiving control commands from the manager. The relationship with the lower layer is made through primitives of the CMIS service. For the higher layer, the primitive of the HMS service are offered. The Meta-MIB presents for the manager above the image of this network, according to the vision of the upper level.

3.2 HMS DATA STRUCTURE

The data structure is shown in Figure 2 and it is composed by a database with all the necessary parameters, such as managers table, agents table, policy table and Meta-MIB. The control box performs the manipulation of data in the tables, it offers the HMS service and implements the HMS protocol, using the CMIS service, specified in section 3.3.

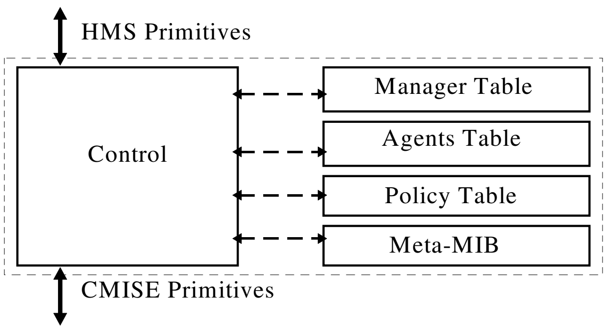

Figure 2 - Data structure of HMS

The Managers Table (TableM) is an address list of the managers devices of upper hierarchical level, in order of priority of connection. If the first manager doesn't answer (it is out of order, for example) contact should be tried with the second manager of the table. The Agents Table (TableA) defines all the objects managed by a specified manager. Through this list, the domains and delegation functions can be accomplished. The Policy Table (TablePolicy) stores all the parameters of the system. We've used as reference some attributes proposed in the TMN standard [11]. The Meta-MIB should store the information of a hypothetical device for the upper level.

3.3 Specification of the Service Offered by HMS Primitives and Protocol

The Domain Management service is based on the specifications showed in the section 2.2.4, and domain management primitives were defined following the suggestions made by SLOMAN et al. [6]. The Policy Management service is based on the specifications showed in the section 2.2.5, and policy management primitives were also defined through the suggestions indicated by SLOMAN et al. [6]. The Delegation Management service is based on the specifications showed in the section 2.2.6. The delegation primitive was based on the concepts presented by GOLDSZMIDT and YEMINI [8] and MOFFETT and SLOMAN [9].

The HMS protocol uses CMIS primitives. As the system accomplishes communications eventually, just when it is necessary some change in the hierarchy or in its parameters, all HMS primitive begin with a association (A-Associate) finishes with a dissociation (A-Release). All the primitives service specification are showed in the Table 1 of the Annex.

4. Project of Hierarchical Management System (HMS)

In this section, we will show the project of the HMS protocol. It is shown a description of the simulation prototype, tests and the methodology used in this project.

4.1 HMS Project Methodology

The project of the Hierarchical Management System (HMS) uses formal description language specification methodology. We have used the Estelle language in our project, by its resources and facilities and also because of the availability of development and verification tools.

In our first specification, we have used the Petri Net model, because its graphical representation is didactic and allows an analysis of the behavior of the protocol control. In this phase, however, it didn't show any data manipulation, which is easier to be handled in the Estelle language. This specification in Petri Nets allowed an initial verification and aided the Estelle specification. However, the Petri Net model was not verified with the appropriate tools, just helping the Estelle specification that was being tested and verified.

4.2 Formal specification

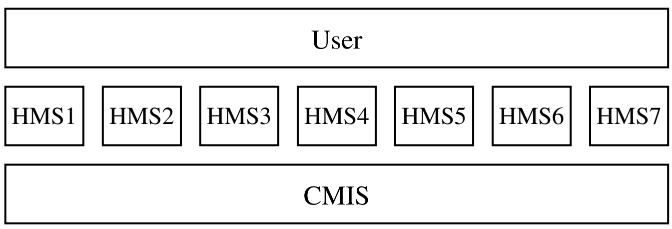

Figure 3 - Simulation model used

The model implemented in Estelle language is shown in Figure 3. The HMS module can be initialized as many times as it is necessary. In this case, we did it seven times. The User module allows the interface with the operator, making easy the SDU (Service Data Unit) assembly, according to service specification. The CMIS module emulates the operation of a layer that offers a management service, using the CMIP protocol. This module just establishes the communication among the several modules HMS, copying the messages from the source module to the destination module .

4.4 THE DEVELOPMENT ENVIRONMENT

The protocols development environment integrated compilation and simulation tools. The project environment used consisted of a Estelle Compiler, which generated two intermediate forms, which were used by the Simulator. It also had a Estelle Code Generator, which mapped directly from the Estelle specification into a executable code. Another separated program used was a Test Sequence Generator, which derived its results from the Finite State Machine model of the system. The details are shown in [12].

4.5 SIMULATION AND TESTS

The simulation was the first step taken towards the validation of the protocol. It basically tried to cover the possible use of all the services of the protocol. The main objective was test the protocol itself, not all the system, that will be done in the future. The tests are shown below.

The first test was the initialization of the hierarchical system through MInitPrin and ManagerInit primitives. The operation was verified using ManagerRead and AgentRead primitives, showed in Figure 4.

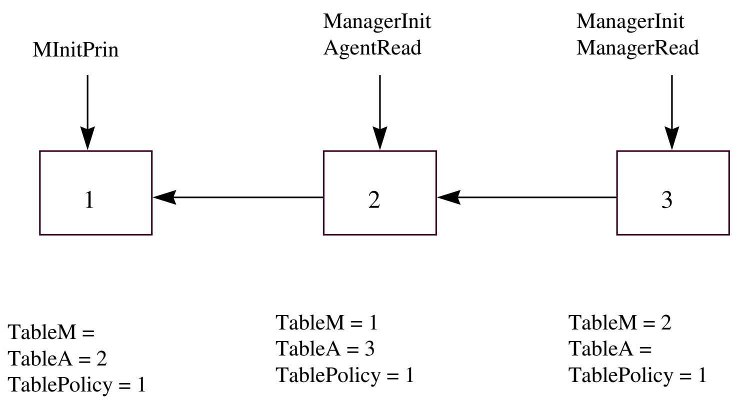

Figure 4 - Initialization of the hierarchical system

The second test was the creation and deletion of an agent, using AgentCreate and AgentElimin primitives, showed in Figure 5.

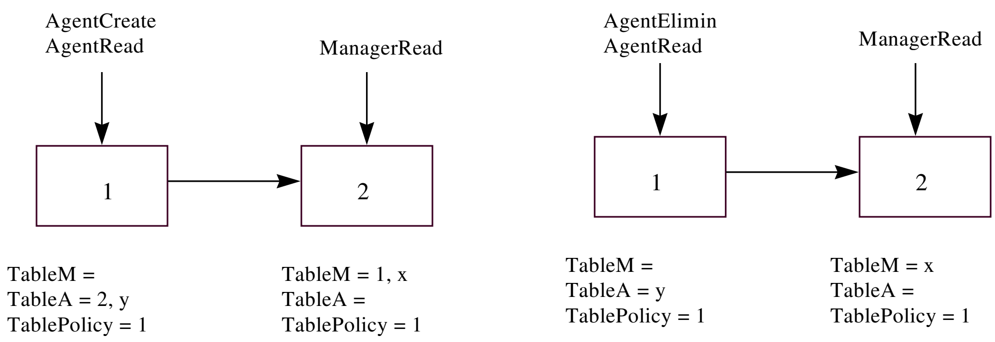

Figure 5 - Creation and Deletion of an agent

The third test was the change of policies, testing the PolicyMRead, PolicyARead, PolicyAChange and PolicyMLoad primitives, showed in Figure 6.

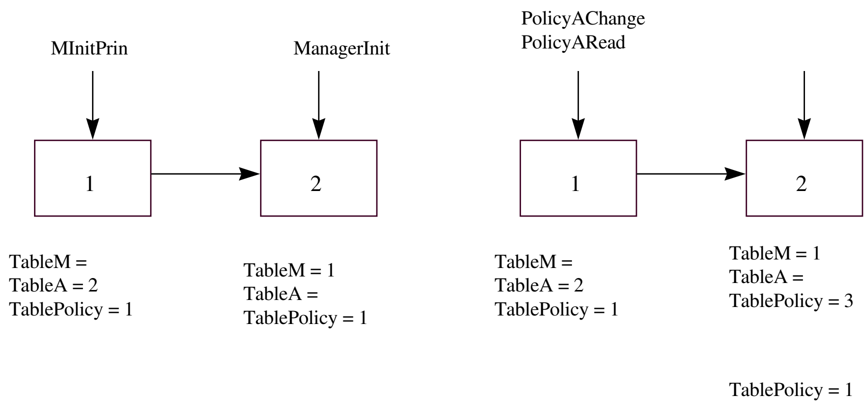

Figure 6 - Policy manipulation

The fourth test simulated a delegation, where Manager2 is delegated by Manager1 to Manager4 using the DelegAct primitive, showed in Figure 7.

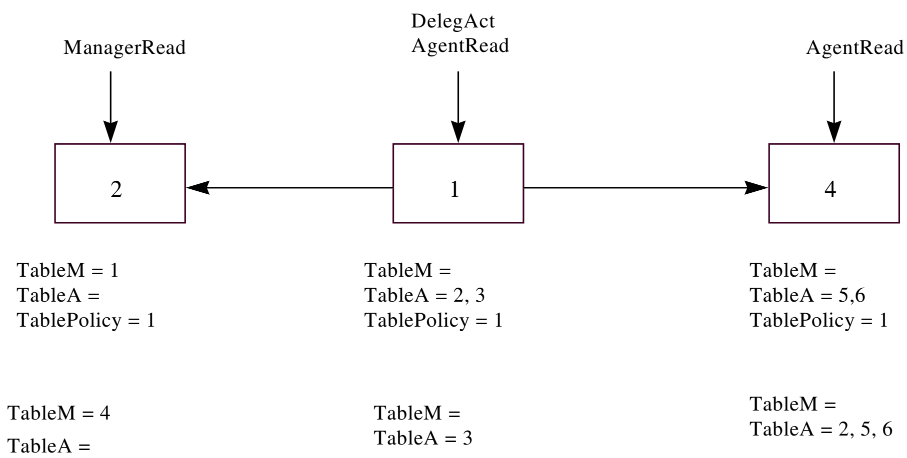

Figure 7 - Delegation test

The following step was to automatically generate the possible transitions through which the protocol should go without deadlocks or exhibition of strange behaviors. In this sense, it was used the Test Sequence Generator, whose sequences were put to test in the simulator once more. Under normal conditions of usage, the results were satisfactory.

The last step was the direct Estelle implementation, in which the code generator acted. In this step, we could make a prototype of the system as in a real case of usage. In this point, the user of the system would work as the operator of the Management Platform, and has no access to the inner part of the system. He just asks for the system to inform him something, and decides what to do to the structure of the network being managed, sending commands through the keyboard.

5. Conclusion

This work showed a proposal for a hierarchical management system, suitable for a large telecommunications network. We have analyzed some proposals found in literature and have used concepts to make a system coherent and flexible. This flexibility of the HMS system, that operates in several elements of the network, does not cause, however, loss of control. The use of policy management guarantees coherence and absence of conflicts in the whole system. Moreover, the possibility of delegation gives the system stability and fault tolerance.

HMS protocol not only can control any number of devices, but also can adapt itself to any change of quantity and localization of managed objects. This work followed all the directives given by ITU and produced an application consistent with the TMN model. The HMS system also offers security resources, because each manager sees only its agents.

A future work could study the policy management technique. This technique reduces the number of messages between a manager and its agents and reduces the execution time of configuration commands. The use of Artificial Intelligence and Fuzzy Logic will allow efficient and fast treatment of policies. Thus, it would be possible to each manager in the system to take most of the decisions alone, reducing the needs for human interference.

6. REFERENCES

[1] ITU - INTERNATIONAL TELECOMMUNICATION UNION. *Telecommunication Management Network Recommendations M.3010: Principles for a Telecommunications Management Network*. ITU, October 1992.

[2] CASE, J., McCLOGHRIE, K., ROSE, M., et al. *Manager-to-Manager Management Information Base. RFC 1451*. USA, April 1993.

[3] BLACK, U. D. *Network Management Standards - SNMP, CMIP, TMN, MIBs and Object Libraries - 2nd Edition*. McGraw-Hill. USA, 1995.

[4] SIEGL, M. R., TRAUSMUTH, G. *Hierarchical Network Management - Concepts and Prototype in SNMPv2*. Technical Report of University of Technology TU-Wien. Austria, May 1995.

[5] SCHALLER, N. H. "A concept for hierarchical, decentralized management of the physical configuration in the Internet", In: *Proceedings of KiVS 95*. Chemnitz, September 20-24, 1995.

[6] SLOMAN, M., MAGEE, K., TWIDLE, K., et. al. "An architecture for managing distributed systems", In: *Proceedings of 4th IEEE Workshop on Future Trends of Distributed Computing Systems*. Lisbon, pp. 40-46, September 22-24, 1993.

[7] WIES, R. "Policy Definition and Classification: Aspects, Criteria and Examples", In: *Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operation and Management*. Toulouse-France, October 10-12, 1994.

[8] GOLDSZMIDT, G., YEMINI, Y.. "Distributed management by delegation", In: *Proceedings of 15th International Conference on Distributed Computing Systems*. New York, June 1995.

[9] MOFFETT, J. D., SLOMAN, M. S. "Delegation of authority", In: *IFIP 2nd International Symposium on Integrated Network Management*. Washington, pp. 595-606, April 1991.

[10] ISO - INTERNATIONAL STANDARDS ORGANIZATION. *9596 Open System Interconnection - Common Management Information Protocol - Part 1: Specification*. ISO/IEC, May 1990.

[11] ITU - INTERNATIONAL TELECOMMUNICATION UNION. *Telecommunication Management Network Recommendations M.3100: Generic Network Information Model*. ITU, October 1992.

[12] FERNANDEZ, M. P. *Protocolo para Gerenciamento Hierárquico de Redes de Telecomunicações*. M.Sc. Thesis of COPPE/UFRJ. Brazil, January 1998.

Annex

Table 1 - List of HMS primitive services and protocol

Primitive	Specification
MInitPrin	Function: Initiate the system defining the main manager in the hierarchic structure and initiates all the necessary tables to start the system. Protocol: It only executes local tables writing and there is no protocol between then.
ManagerInit	Function: Includes a manager in the hierarchic structure of the system. It sends a message for its manager who includes it in the hierarchy and receives all the necessary tables for its operation in the structure. Protocol: After the association is established the upper manager gives TableM and TablePolicy tables using the M-Get primitive. After this, the closing process will be executed.
ManagerRead	Function: Reads the managers list stored in TableM Protocol: It only executes local tables reading and there is no protocol between then.
AgentRead	Function: Reads the agents list stored in TableA Protocol: It only executes local tables reading and there is no protocol between then.
AgentCreate	Function: It includes an agent in the local agents table (TableA) and includes a manager in the remote manager table (TableM). Protocol: After making the association, the initiating manager sends information for the other using M-Set primitive, modifying remoteTableM table. Then it modifies its own TableA table and makes the communication closing.
AgentElimin	Function: It removes an agent of the local table of agents (TableA) and removes a manager from remote manager's table (TableM). Protocol: After making the association, the initiating manager sends information for the other using M-Set primitive, modifying remoteTableM table. Then it modifies its own TableA table and makes the communication closing.
PolicyMRead	Function: It reads local policy table. Protocol: It only executes local tables reading and there is no protocol between then.
PolicyARead	Function: It reads policy table from an agent. Protocol: After establishing the association with the agent, gets information from its TablePolicy using M-Get primitive and closes the communication.
PolicyAChange	Function: It modifies Agent's policy table. Protocol: After establishing the association with the agent, changes the agent's TablePolicy information using M-Set primitive and closes the communication.
PolicyMLoad	Function: It asks upper manager to update local policy table (TablePolicy). Protocol: After establishing the association with the manager, changes its own TablePolicy using M-Get primitive and closes the communication.
DelegAct	Function: It requests an authorization to delegate an agent to a manager. If affirmative it modifies its TableA, the TableA of the manager called and TableM of the agent who will be delegated. Protocol: The Manager establishes an association with the manager called and asks for authorization to delegate an agent using M-Action primitive. If the answer is affirmative the manager called will change the TableA as to include the new agent and the communication will be closed. Then, it establishes an association with the Agent in order to change its TableM with the new Manager using M-Action primitive. The communication will be closed and the first Manager will change its own TableA, excluding the Agent.