# An Architectural Approach for Integrated Network and Systems Management


Raouf Boutaba[1] and Simon Znaty[2]

[1] Laboratoire PRISM, Université de Versailles St-Quentin-en-Yvelines,
45, avenue des Etats-Unis, 78035 Versailles cedex, FRANCE
Tel : 33 1 39 25 40 42, Fax : 33 1 39 25 40 57
E-mail : Raouf.Boutaba@prism.uvsq.fr

[2] Swiss Federal Institute of Technology, Telecommunications Laboratory.
CH-1015 Lausanne, SWITZERLAND.
Tel : 41 21 693 56 13, Fax : 41 21 693 26 83
E-mail : znaty@tcom.epfl.ch



## Abstract

Today's enterprises are accepting networked systems as a fundamental part of their information technology strategy. The constant growth in quantity and quality of networked systems and the thereby arising problems concerning complexity, heterogeneity and diversity of components in a multi-vendor environment require a sophisticated management of resources. Increasingly the automation of such management is being demanded.
In this paper we introduce an architecture for the integrated management of all resources in a networked system, i.e. application, system and network resources. The architecture uses domains as flexible and pragmatic means of grouping resources and of specifying management responsibility and authority boundaries. It maintains a clear distinction between management objectives and the resources being managed in order to provide an integrated view of the various tasks of management as well as an integrated and uniform view of the distributed and heterogeneous managed environment. The uniform management model on which the architecture is based is expressive enough to capture the full richness of management structures and policies both within enterprises and between them. It allows for recursive and generic structuring, we consider as the basis for management activity automation.
As an example, we apply our architectural concepts to structure the management of a high speed multi-network (ATM, DQDB, FDDI). Emphasis lies on an automated Quality of Service management in the FDDI management domain.


## Introduction

Today's enterprises are accepting distributed systems (information processing systems supported by communication networks) as a fundamental part of their information technology strategy. The constant growth in quantity and quality of networked systems and the thereby arising problems concerning complexity, heterogeneity and diversity of components in a multi-vendor environment require a high quality management. Increasingly the automation of such management is being demanded.

Management quality comes through good organization. This requires that the networked system environment and its management tasks are appropriately structured. Therefore, we have defined a management architecture as a guideline for building and organizing management systems. Two structuring principles underlie the realization of our architecture. The first is the use of domains as a means for grouping resources determined by management need (e.g. a domain applying a specific management policy). The dependencies between the emerging domains reflects both hierarchical interactions (e.g. control of resources, authority delegation to subordinate managers) as well as peer-to-peer interactions (e.g. negotiations between peer managers to prevent/resolve management conflicts). The second structuring principle is to separate management policies from the resources and activities being managed. This separation allows, on one hand, managers to have an abstract and uniform view of the managed resources hiding these latter's heterogeneity and, on the other hand, implementation of a uniform and generic model for the management activity.

Defining and applying such a management model is the key step towards management automation which constitutes the ultimate aim of this work. However, the semantics of management are not yet adequately understood. Besides a very coarse model describing management as the activity of gathering information and exerting control, the complex task of management is usually built into management applications. Such an approach hinders integration and open cooperation. In addition, in order to provide an automated management activity, it is necessary to be able to dynamically specify high level management policies and to turn them automatically into low level control commands to be executed on managed resources. The complexity of the tasks of making and interpreting (informal) management policies make difficult such automation. These tasks are often the responsibility of human managers.

This paper discusses the way management is performed and identifies the different kinds of management information to be manipulated. It proposes a model which describes the different steps of the management task and explores how the model may be applied within our domain based management architecture. More precisely, three kinds of management information are defined : goals, policies and plans. The model starts with high level goals (e.g., as input from a superior authority) and ends up with management plans stating exactly what is to be done on the occurrence of certain situations. Management policies are introduced as intermediaries between (abstract) management goals and (executable) management plans. They are general statements about how management goals will be achieved, and are used to ease decision making. Adhering to this model enables the dynamic change of goals, policies and plans which are currently very often hard-coded into management applications. Furthermore, parts of the management task can be formalized and hence automated.

This paper is structured in two parts. Part I is composed of five sections. In Section 1, we introduce our architectural principles for structuring the management of large scale networked systems highlighting the hierarchical nature of management and show how management authority is delegated between managers. Section 2 discusses the structuring of our management domains into a managing part and a managed one. Section 3 focuses on the managing part of the domain (called the kernel). It aims to explore the refinement of the management activity and identifies the corresponding computational units. Section 4 addresses in more details the executive which is the component of the kernel responsible for observing resources and subordinate managers as well as exerting control on them. In Section 5, we outline an approach to the provision of an automated policy and plan making, by means of a more simple representation of management goals, policies, plans and their inter-relationships. In the second part of this paper, we apply our concepts to structure the management of a high speed multi-network (ATM, DQDB, FDDI). A multi-network is an assemblage of communication equipment and software that enable the user to consider the overall network as one resource only. Today's principal problem is the administration of these heterogeneous systems (multi-network) after being that of networks architecture and their interconnection. As multi-networks grow in size at a rapid pace, the various components and multi-network users interact in increasingly complex ways. These complex interactions imply the need for intelligent, automated, efficient, and integrated management. Our emphasis lies on QoS management in the FDDI management domain. The work described in this paper has been partly conducted in the scope of the Esprit II project DOMAINS whose basic concepts are presented in [1].

# I. The Architectural Concepts

## I.1. Management Architecture

The structure of a management system for a networked system has to reflect corporate requirements in the following sense : companies have to structure the tasks and responsibilities in order to apply certain management policy in well-defined areas. These areas may have hierarchical or peer-to-peer relationships. To allow for a corresponding feature in the overall management system, to reduce complexity, and to allow for partitioning and distribution of the overall management task, we have defined a management architecture based on two structuring principles :

- P1: Use of *domains* as organizational units;
- P2: Make a clear distinction between management policies and objectives and the resources and activities being managed.

The first structuring principle (P1) is concerned with the construction of the management system identifying this one organizational units and their relationships. Indeed, management complexity is often reduced by structuring and organizing the management system into subsystems and by distributing management responsibilities over these subsystems. In this perspective, principle P1 introduces the domain concept as the building block. A domain is an area of authority and responsibility and provides a flexible means for grouping resources for management purposes. It has been used by a number of groups in the USA for security purposes [2], [3], [4]. They are also used by other research groups (e.g. in the Domino Project [5]), [6] and standards [7] for explicit grouping of resources. While in these works domains are either managers or managed, our domains comprise a set of managed resources but also encapsulate the components performing management. This choice reflects our hierarchical design of the management process ([8]). It allows for less flexibility when building the management system but allows the management activity to be applied uniformly at all levels of the hierarchy and thus makes it easier to automate.

The whole management system is then logically constructed in a hierarchical domain structure where low level domains provide their services to those of the upper layers. Management complexity is reduced by separating management concerns and by refining management tasks through the different levels of the domains' hierarchy. Management tasks are provided by individual domains which cooperate to achieve the global objectives. Another important advantage of this domain based structuring is to simplify the evolution of the management system, e.g., by adding a new domain or deleting an existing one without disruption of the entire management system.

As stated above, our domain comprises a management part (a manager) as well as a managed one (a set of managed resources). The managing part manages resources according to a given policy or a set of harmonized policies. Therefore, managers and resources may be related to construct domains according to different criteria that may be relative to the contained managed resources or to the contained management functions and policies. Examples of such criteria include :

- *Management function criterion*: This type of domain applies to the contained managed resources a particular management function or a set of management functions (e.g., the standard functions defined by the ISO or a meta-management function). Examples of such domains include: - a maintenance manager with the concerned resources; - a high level manager (an administrator) performing a meta-management task such as system structuring, policy conflicts resolution or recovering from management failures.

- *Policy criterion*: This is used to separate out management concerns and simplify the task of setting policies. It applies to the managed resources the same management policy (e.g., a security policy).
- *Organizational criterion*: This defines boundaries reflecting the organizational structure of a given system or a sub-system, e.g., a research group within a university laboratory.
- *Location criterion*: This type of domain defines a geographical boundary containing co-located physical and logical resources, e.g. a LAN with a set of workstations.
- *Ownership criterion*: This is useful for security management and deals with access control to the contained resources, e.g., a set of files that are private to a particular user.
- *Managed Resources type criterion*: This groups managed resources that belong to the same vendor (SUN workstations, PCs, etc.) or those that are conform to a particular standard, e.g. ISO conformant inter-networked systems. This last domain type may be self managed.
- *Managed Resources functionality criterion*: This groups the components that have the same functionality, an equivalent one or cooperate together to provide a particular service. Examples of such domains include: - all the printers within an organization; - Electronic mail service including databases, user interfaces, mailboxes and an administrator for service configuration and user registration.

These structuring criteria and others may be used simultaneously to build the same domain based management system (first structuring principle : P1). They must be applied judiciously to make easier the management of the overall system and to prevent/categorize conflict situations.

The second structuring principle (P2) is concerned with the domain's internal structure identifying and refining the computational units and their interworking. Our goal is to ensure a uniform and automated management of all resources and activities in a networked system. These last range from hardware resources such as lines and switches to software resources such as processes and databases. Therefore, principle P2 has been defined to separate the management activity from the resources that are subject to management. This allows us to define a uniform model for the management activity in order to automate it.

Each management domain has to realize the common management model where a manager *monitors* and *controls* resources to meet management objectives (we call *goals*). Monitoring allows one to get knowledge about the resource's behaviour and to check their consistency with respect to the goals. A management by delegation mechanism (cf. [10]) provides the overall management task. Indeed, domains may be structured so as to exhibit a delegated management role with respect to another domain and this delegation may be applied recursively through the different levels of the management hierarchy (it is depicted by Figure 1 at levels N and N-1). The delegation relationship and structuring principle is to be understood as follows. A managed resource within a domain can be as simple as a real resource (e.g. a device) or as complex as a management system itself (a domain of a lower level). This subordinate domain contains again a manager which may exercise delegated control over its set of managed resources. In addition, the manager *reports* to its superiors about its management particularly when it fails to achieve the goal.

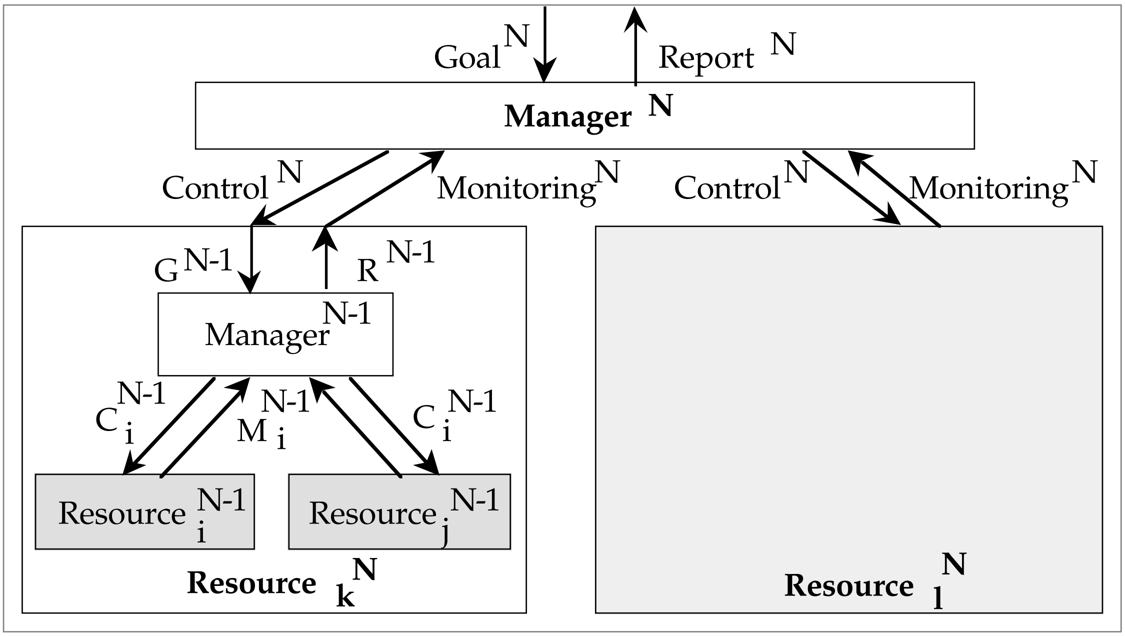



**Figure 1 : Recursive management model**

Applying principles P1 and P2 leads to a management system consisting of multiple coexisting management domains that can be autonomous or interacting in hierarchical or peer-to-peer relationships. Hierarchical relationships make the management system hierarchy where upper layer domains delegate certain management tasks (as management goals) to their subordinates to be performed on these managed resources. Peer-to-peer interactions take place in case of overlapping between domains (i.e., domains sharing the management of one (or several) resource(s)). A peer-to-peer cooperation between the overlapping domains managers may be necessary in order to synchronize their respective management and to provide total control of the shared resources (i.e., an optimized and non-conflicting management). For more detail about management relationships and domain dependencies cf. [9].

The degree of overlap greatly depends on the adopted structuring criteria. In case of location based partitioning, overlapping is not frequent and is limited to geographical boundaries, for example, shared management of a gateway interconnecting two LANs if each of these corresponds to a domain. Overlapping occurs more frequently when applying the manager functionality criterion. This may lead to conflict as, for instance, a workstation put out of service by the maintenance domain is unavailable in the scheduling domain if these two domains share the management of this workstation.

Now, having introduced our management architecture we want to develop, in the following section, the internal structure of the domain so as to reflect our second architectural principle.

## I.2. Domain Internal Structure

As stated in the previous section our domain consists of a set of resources and a management part applying a certain policy or an aspect of a policy. The management part of a domain is called the *Domain Management System* ("DMS"). We identify two basic parts of a DMS : the abstract representation of the managed resources (called the *Shield*) and the managing part performing management actions. The latter part we term the *Kernel*.

The kernel is the access point to a domain and contains all activities which belong to the managing part. It performs control actions over the managed resources through their representation in the shield. Managed resources are called *Target Resources*. Each target resource is either a real resource (e.g., a switch, a printer, etc.) or another domain of a lower level in the management hierarchy (see Figure 2). The shield is an interface component introduced for openness and reusability purposes; it hides the resource's functional interface and gives an abstract view of the management interface (cf. the management information model in OSI management). The shield is thus the domain's component which provides the separation between the management activity and the resources being managed (principle P2).

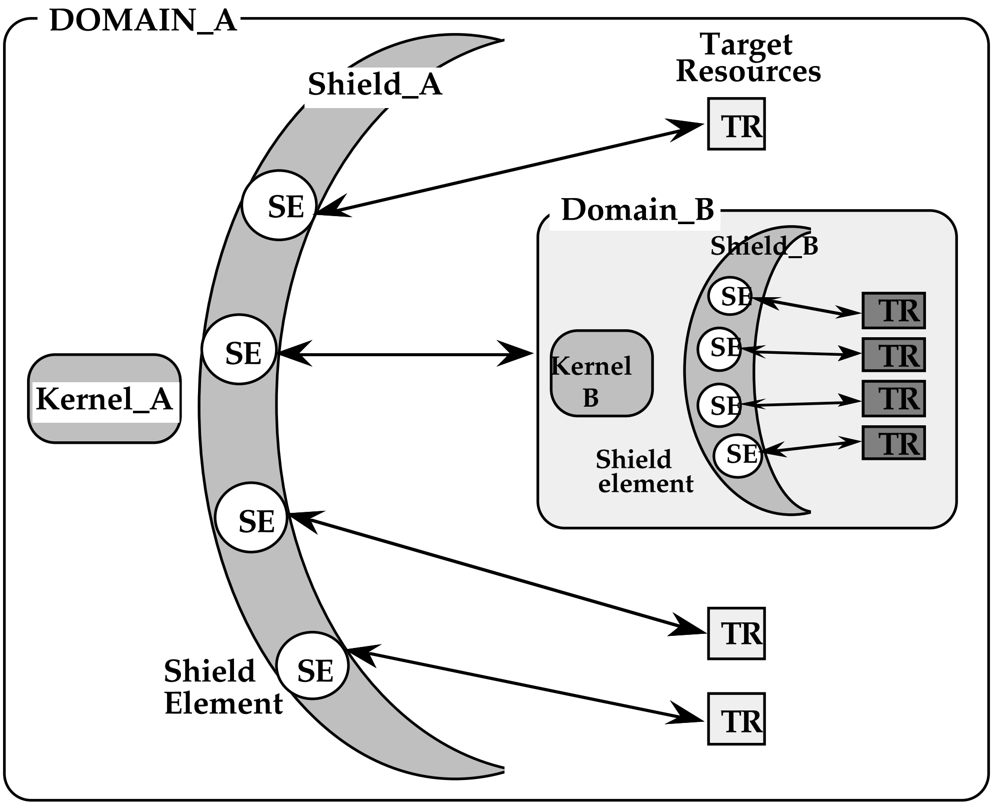



**Figure 2. Internal structure of a domain**

The shield has no autonomous management activities. It presents a *uniform*, *selective* and *abstract* view of the domain's target resources to the managing kernel. Indeed, the kernel may need to have a uniform view of a number of different resources (e.g. different proprietary management interfaces of the same type of resource can be unified). The abstract view allows hiding of irrelevant details from the kernel and the selective view allows restriction of kernel access to the part of the resource interface relevant to the management objectives of that kernel. Each target resource is represented by a shield element within the shield. Some functions of the shield element can be executed at compile time such as name translation but a shield element might also perform run-time activities such as protocol conversion when interfacing with a real resource. Indeed, one may request the shield element to:

- Forward kernel messages to the target where the action requested by the kernel is to be performed. This may include protocol transformation.
- Limit kernel access to those parts of the target interface that are visible in the selective view provided by the Shield.

- Ensure that notifications are sent to the kernel when it expects them, and provides support for the retrieval of information from the Target when decided by the kernel.
- Ensure time value conversion as well as explicit synchronization between kernel and targets.
- Ensure that attribute types of the information available within the target match the kernel's expectation. This may lead to simple scalar type translation indeed even complex aggregation.

The shield concept offers two advantages : the access to the shield does not have to be changed if a resource is exchanged (e.g., two printers); and the kernel does not have to know about details of accessing a resource.

In the following, we want to model delegation of management and clarify the internal structure of the managing part, i.e., the kernel. We introduce the terms "management goal", "management policy", and "management plan" to describe the information a kernel deals with. The input of a kernel is modelled as a set of goals which are a high level specification of management objectives. These are turned into policies which restrict the way the goals are achieved. From these policies plans are derived which state what exactly is to be done. In the subsequent section we give more thorough explanation of these terms and specify the kernel components using the corresponding information.

## I.3. The Kernel

The kernel receives management goals from higher level kernels. A management goal is a statement about what is to be achieved. The kernel achieves the management goals by taking actions which are single management operations. Actions available to the kernel include issuing instructions (a command to do something) to the resources being managed or to other managers involved in managing the resource, making reports (to the issuers of the management goals), performing internal operations, and possibly others. The kernel performs control actions over the managed resources according to a management plan which is a procedure of actions which can be deterministically evaluated at the time it is to be executed. A kernel makes management plans in order to deal with :

- a new set of management goals,
- a situation (a description of a pattern of observations) which requires some action(s) to be taken in order to satisfy management goals.

Because making management plans from a set of management goals can be a very complicated task, the concept of management policy is introduced as an intermediate step between management goals and management plans. Management policies are general statements about how the kernel will achieve the management goals, and are used to ease decision making by restricting the set of solutions to a problem from the range given by the management goals to a more easily handled size.

Management plans derived from management policies may be executed immediately to achieve some of the management goals (e.g., "turn on accounting"), or may be stored for execution in response to some situation as part of a persistent management goal (e.g., "if response time exceeds 50 ms, reroute"). In the latter case, the management plan consists of two parts : the first part is a description of the situation and the second part is the action to be taken.

The management activity within the kernel consists then of three distinct phases :

- making management policies from management goals;
- deriving management plans from management policies;
- executing management plans.

Sometimes the decision making is quite simple, resulting in a management policy which is little more than a restatement of the management goal, or a management plan which is identical to the management policy. For example, if a management policy states a course of action which the manager is able to perform directly, then the management plan is the same as the management policy. If a management goal is to be achieved by passing it on to another domain, the management policy may simply repeat the goal, adding that it is to be passed on. However, in the general case this is not true, and when the management goals are long-lived it is especially important to store the management plans (as they have to be executed over a long time) and management policies (as they affect many other operations of the kernel).

The identified management phases are respectively processed and supervised by the kernel active entities illustrated in Figure 3. These are :

- the Policy Maker,
- the Planner,
- the Executive.

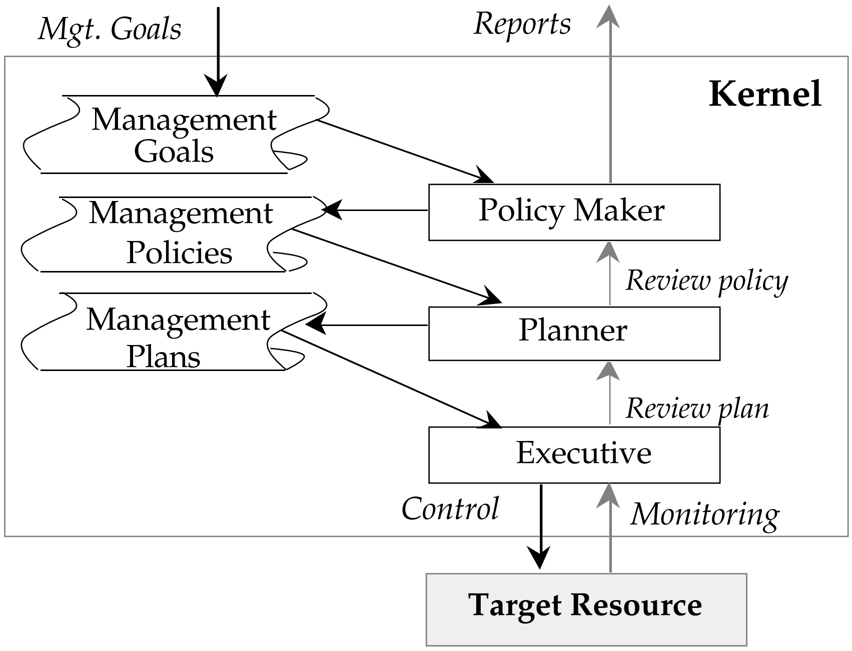



**Figure 3 : Kernel Internal Structure**

This structuring of the management activity within the kernel enables the dynamic change of goals, policies, and plans which are currently very often hard-coded into management applications. In order to avoid unnecessary overhead, kernels may be configured in such a way that only useful functionality is included allowing for mighty ones with e.g., policy derivation capabilities down to small ones with plan execution capability. In the following we discuss the full scale configuration of a management kernel; we distinguish two modes of operation :

- *Proactive management* stimulated by arrival of goals from higher level managers and,
- *Reactive management* on detection of events or arrival of notifications from managed resources and/or lower level kernels.

In the proactive management process, the Policy Maker has to derive management policies for each received management goal immediately. The obtained management policies are used, as input, by the Planner to make plans. The obtained plans may be executed immediately to achieve some of the management goals, or may be stored for execution in response to some situation as part of a continuing goal. The feed back depicted by the dashed arrows in figure 3 may be a positive as well as a negative confirmation, e.g., "objective is achieved, current attribute values are...", or "objective cannot be met, because...". Objectives in these examples are goals, policies and plans respectively. For example, when the executive is unable to execute a job, a message is sent to the Planner asking it to review the plan or to make an alternative one which can be executed. Thus feedback serves for synchronization purposes as well as for monitoring the effectiveness of proactive management.

Reactive management consists of executing control actions (according to a stored plan) when a particular situation is detected at the managed resource's level. When a situation occurs for which no management plan exists, a new management plan is made from existing management policies by the Planner. If no (suitable) management policy exists, a new management policy is to be derived from existing management goals. If no relevant management goal exists, the action taken by the Policy Maker depends on its own meta-policy, e.g., set a management policy using default management goals, do nothing, issue error report, etc.

The Policy Maker and the Planner must have information about the capabilities of the resources to be managed and the resources that can be used for managing in order for them to make decisions. Now, having introduced these components of the kernel we want to give, in the following, a more detailed explanation of the Executive. This component is located at the interface between the kernel and the shield, so that its activities show the interworking between hierarchical managers or between managers and resources.

### I.4. The Executive

The main task of the Executive is to execute jobs which are one of two parts of the management plans introduced in the previous section. The other part consists of situation specifications. To provide its functionality the Executive may be logically structured into three distinct components (see Figure 4) :

- Execution Engine;
- Situation Matcher;
- Observer.

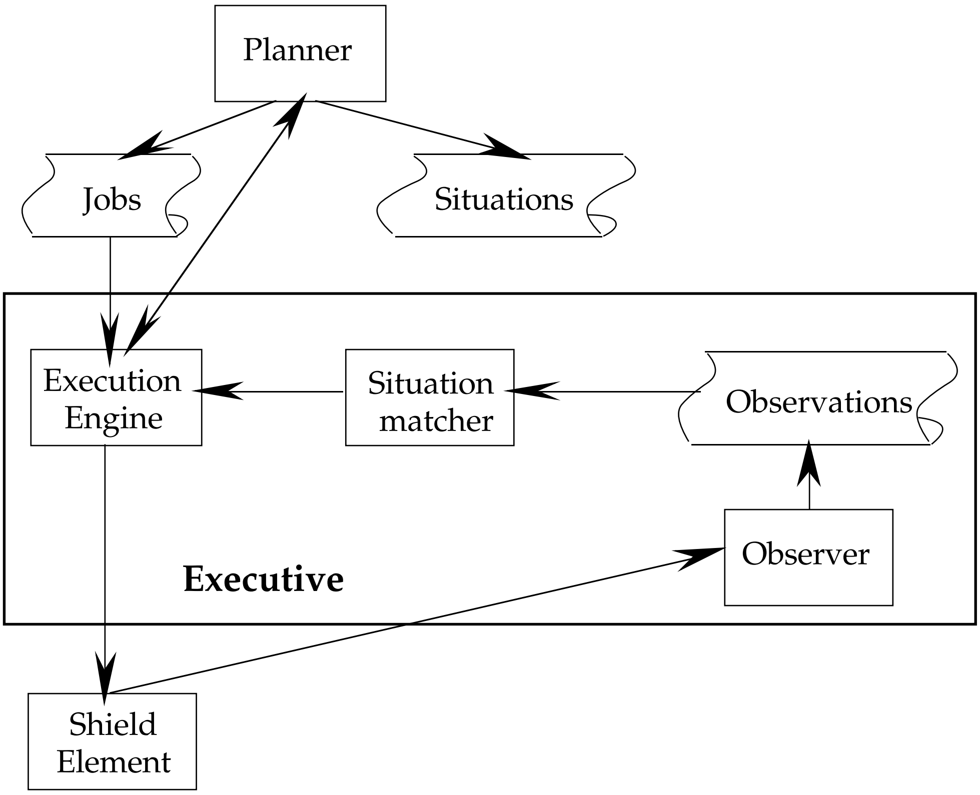



**Figure 4: The Executive Internal Structure**

Jobs are executed by the Execution Engine, as soon as they are created by the Planner, or in response to some situation. The latter case is possible by matching the observed state of the system with stored situation specifications described in terms of patterns of observations. The observations are provided by the Observer. Jobs are represented by algorithmic blocks and their execution results in a sequence of instructions being sent to shield elements that represent resources or managers of resources.

The Execution Engine is invoked by the Planner or the Situation Matcher to execute jobs. The planner supplies the store of situations each of which is linked to a corresponding job which is to be executed. When the Situation Matcher detects a situation it passes an identifier of the corresponding job to the Execution Engine together with any parameters relating to the job. Indeed, jobs invoked by the Situation Matcher may be parameterized to allow one job to work in different but similar circumstances. For example, a job may be parameterized in terms of which object in a set of similar objects must be changed, or to allow different instructions to be sent depending on the observations such as the degree of failure against some performance criterion. The Situation Matcher computes and supplies the parameters. It may be possible that the parameters cannot be computed. In this case the job may fail due to a special error, but it may be possible for the job to work anyway, using default values for example.

Situations are defined in terms of combinations of the values of observables (an attribute of some object which has a value which can be measured) and times of observations. The language used to express situations must be able to represent observables and observations and provide operators for obtaining :

- the value associated with an observation,
- the time an observation was made,

- the value of an observable at a particular time,
- a reference to a particular observation from an observable.

Such language must also provide mathematical operators for combining values and times. Internal variables are also needed (cf. [11], [12]).

### I.5. Representation of Goals, Plans and Policies

The ultimate aim of this work is to automate the management process as modelled in section 3. This assumes a formal specification of the different kinds of information processed within the kernel (i.e., goals, policies and plans) as well as tools for automatic derivation of policies from goals and plans from policies. However, even if the second management phase (i.e., deriving management plans from management policies) can be performed automatically by a machine, the first phase (i.e., making management policy from goals) can only be performed by a human or by a sophisticated program without time constraints. This is due to the complexity of the policy concept (not yet formally specified).

The research community is only now beginning to address the issue of how to specify management policy. For example, in the Domino project ([13], [14]) management policies are implemented as system objects, but limited to access rules. The Pythagoras project (cf. [15]) is also concerned with modelling policies in order to create a database of the policies of an organization. A policy in pythagoras is a right or a responsibility declared so as to prompt action which conforms with an intention. The database is queried by users in order to ascertain what policies exist in a specified subject area. However, the system does not interpret or constrain the contents of policies. More recent works look at deontic logics, the logics of normative systems ([16], [17]), as candidates for expressing policies. Indeed, these logics have operators which denote both "obligation" and "permission", either of state or actions. Obligation and permission are two preconditions that must be satisfied for a manager to perform management actions. The first one specifies the goals of the organization and how they are to be achieved (e.g. set constraints limiting the way in which the goals are to be achieved). The second one allocates (give access authorization to) the resources which are needed to carry out the goals. Both of obligation and permission policies are often used in hierarchical fashion acquired by a manager through delegated goals and authority. This suggests many levels of policy from high level objectives of an enterprise to low-level management actions to be performed on the underlying information technology. In this perspective, Moffet & al. [18] have explored further the refinement of general high level policies into a number of more specific policies to form a policy hierarchy. They made a distinction between imperative and authority policies which may be equated respectively with obligation and permission policies discussed so far. Furthermore, expert system support for the analysis of policy hierarchies is provided allowing determination of whether lower level policies satisfy the higher level ones. The system does not provide tools to automate policy generation from abstract high level objectives, although this might be considered as a future development.

Most of these works made no distinction between management objectives, policies and plans. They faithfully adopted the common definition of policies as the plans of an organization to meet its objectives. We make such a distinction by splitting up the information structures relevant for management into goals, policies and plans. As discussed in section 3, this allows us to handle both abstract high level objectives and low level control commands and to reduce the gap between them by means of automated policies and plan derivation and execution.

In the remainder of this section, we want to show the viability of our automated three-phases management model. For that purpose and in the scope of our high speed multi-network management (see following sections), the full complexity of the Goals-Policies-Plans-Actions model will not be demonstrated, but a scheme has been devised for implementing each of these steps in a limited scenario. This scheme involves designing each managing kernel in such a way that only a limited and well defined set of goals can be specified, from which a set of policies can be determined automatically and then turned into a plan of actions by applying *Policy Predicates* to a predetermined set of plans in the form of a *Plan Template*.

For this purpose, two approaches are combined for making policies : the bottom up and the top down approach whether policies are determined from plans or derived from goals.

In the bottom up approach, the designer of a kernel must investigate, as a first step, what possible actions are to be performed on the resources managed by this kernel. These actions are then put together to form a sensible and coherent set of sequences of actions (i.e., the plan template). A plan template consists of a tree of action sequences with state and policy decisions at each branch. It is represented diagrammatically as shown in Figure 5 where valid elements are :

- Triggering events (Incoming messages and notifications),
- Send Message Actions (Outgoing messages),
- Other actions,
- State-dependent decisions,
- Policy-based decisions.

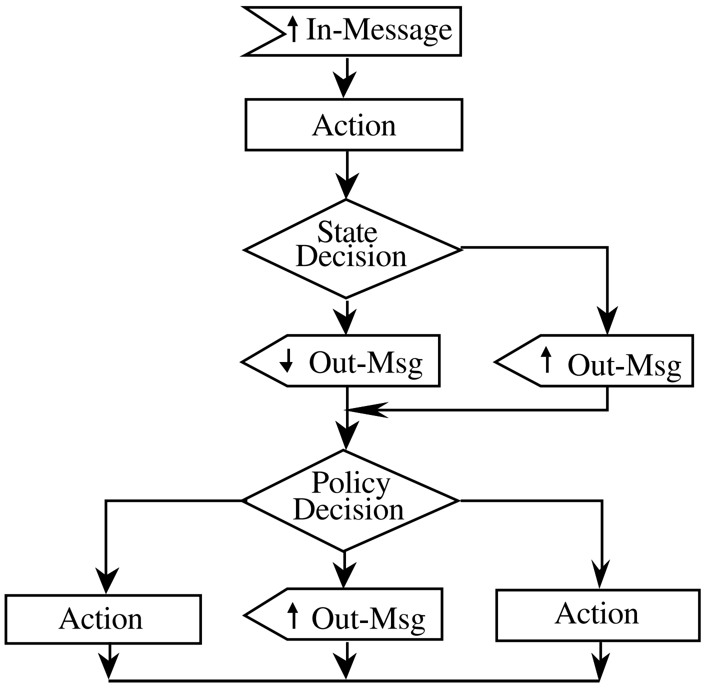



**Figure 5 : A plan Template**

When constructing the plan template, the designer must decide which branches are controlled by policy decisions and which by decisions based only on the managed system state. Having done this, the designer must define what policies affect the policy decisions and how such a decision will be reached. These are defined in conjunction with the second approach.

In the top down approach, the designer must firstly, decide what kind of goals will be desired to be given to this managing kernel and specify which ones will be implemented. From these goals, and in conjunction with the policy decisions derived in the bottom up approach, the designer must decide the composition of policies. Policies are expressed in the form of a policy predicate, a deterministic expression whose evaluation provides the result of a policy decision. For example, a manager may have to decide between fixing a broken connection or reconfiguring the connection out of use. The factors involved in such a decision may include the cost of the repair in terms of money, time, perceived inconvenience, permanence of solution, etc. The policy predicate will reflect these factors and how much importance the designer places on them.

In the top down approach, policies have a goal-like flavour : "don't spend more than 1000 accounting units per day"; "Efficiency has priority 5, reliability has priority 12". By contrast, in the bottom up approach policies have a plan-like flavour : "Always send a notification to your manager when performance is degraded"; "If connection breaks, repair it". The designer has to find a representation which is an amalgamation of these two views.

There are two possibilities for the nature of a policy : it can either specify a potentially complex calculation to determine a policy decision, or it can specify the result of such calculation. However, the latter case can only apply to decisions that can be made statically. It seems likely that many policies will include some state-dependent elements, making them dynamic policies. Thus we represent policy predicates as explicit calculations rather than as simple result look-ups. This has the further advantage that purely state-based decisions can be implemented in the same way.

## II. Application to a High Speed Multi-Network

The remainder of this paper focuses on the application of our concepts to the management of a high speed multi-network (ATM, DQDB, FDDI). In particular, we apply this simplified goals-policies-plans-actions scheme for the automation of Quality of Service management in the FDDI domain.

### II.1. The Multi-Network Environment

For the application of the proposed concepts, we consider a high speed multi-network (FDDI, DQDB, ATM, Token Ring, Ethernet) simulator (Figure 6) whose development has been carried out at the network department of Télécom Paris. The obtained tool called S.A.R.I. (**S**imulateur d'**A**dministration de **R**éseaux **I**nterconnectés) consists of two parts:

- A simulator part, which models (in terms of structure, behavior and operations) all the network components of a high speed multi-network (ATM, DQDB, FDDI) in order to observe their functional behavior and thus to provide a basis for the study of their management.
- A demonstrator part whose objective is to effectively show the feasibility of the management of the S.A.R.I. Simulator part.

The implementation has been supported by the Eiffel object oriented language. Both Network and MAC interconnection levels have been considered, involving respectively bridges and routers. This object-oriented multi-network consists of a DQDB MAN as backbone for FDDI, Ethernet and Token Ring interconnection through bridges or routers, and a B-ISDN network. For our scenario, we assume internetworking at the MAC level and emphasize the application of the proposed "Domain" approach. We first structure the simulator network configuration into domains.

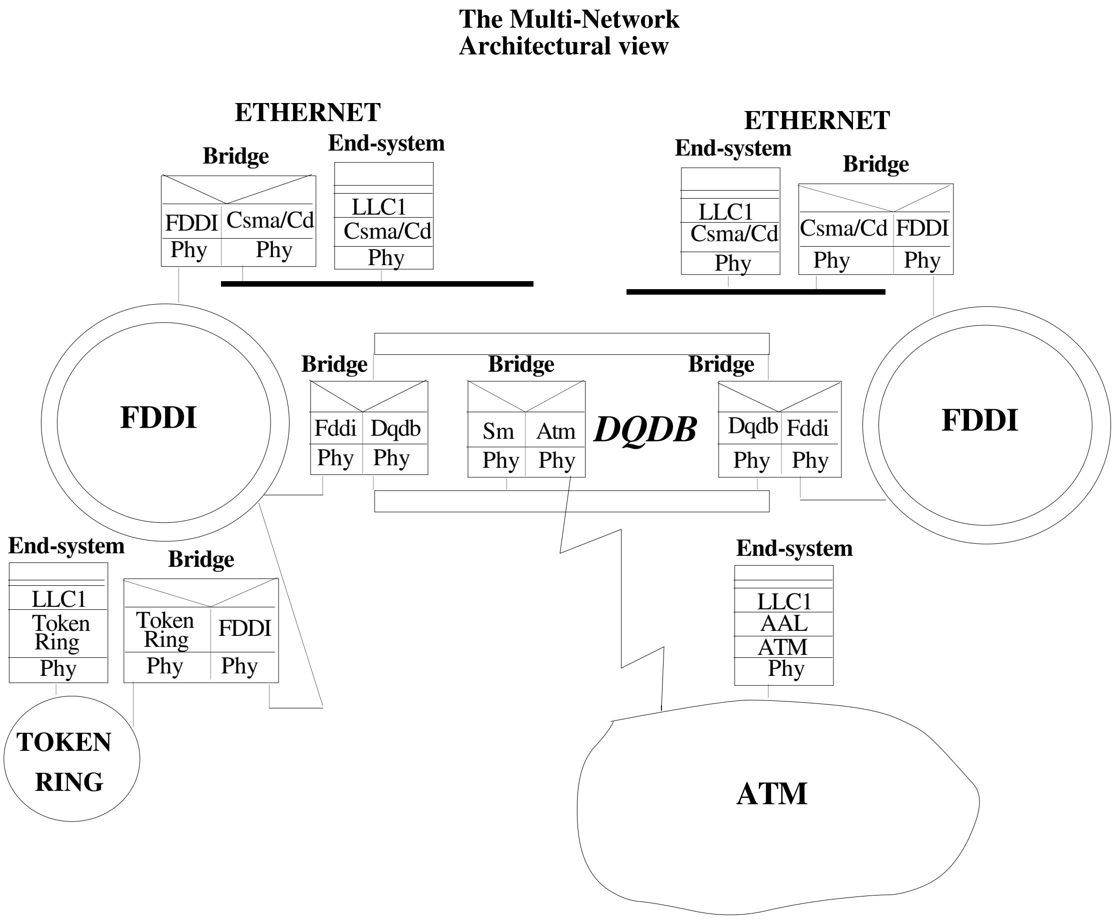



**Figure 6: the architecture of the multi-network**

## II.2. The Multi-Network Domains

In the following, we present the rules that have been applied for defining the hierarchy of domains within our multi-network as illustrated in Figure 7.

Public and private domains have been separated according to the *policy* criterion. The private domain groups the FDDI high speed backbone subnetwork and the medium performance LANs (Ethernet, Token-Ring) it interconnects. The public domain is composed of the DQDB and ATM subdomains which correspond to a public MAN and a public WAN respectively. Indeed, these two domains do not share the same responsibilities, objectives, and requirements. For example, accounting is much more crucial for public than private domains. On the contrary, the latter are more concerned with security constraints.
Within a public or private domain, we apply the *location rule* to separate the different subnetworks in domains. The private domain has been subdivided into several subdomains corresponding to the subnetworks it contains (Ethernet, Token Ring and FDDI). The public domain applies the same principle. It encapsulates two subdomains namely DQDB and ATM domains concerned with the DQDB and ATM networks respectively.
Note that the FDDI domain (N) enables the monitoring of the global FDDI subnetwork behavior and its efficient management in a multi-network environment while the FDDI management

standard only focuses on the management of an FDDI station. This remark also applies to DQDB and ATM.

Moreover a structuring according to the *organizational criterion* creates a new domain which contains Token Ring and Ethernet subnetworks that belong to the same organizational structure for example.

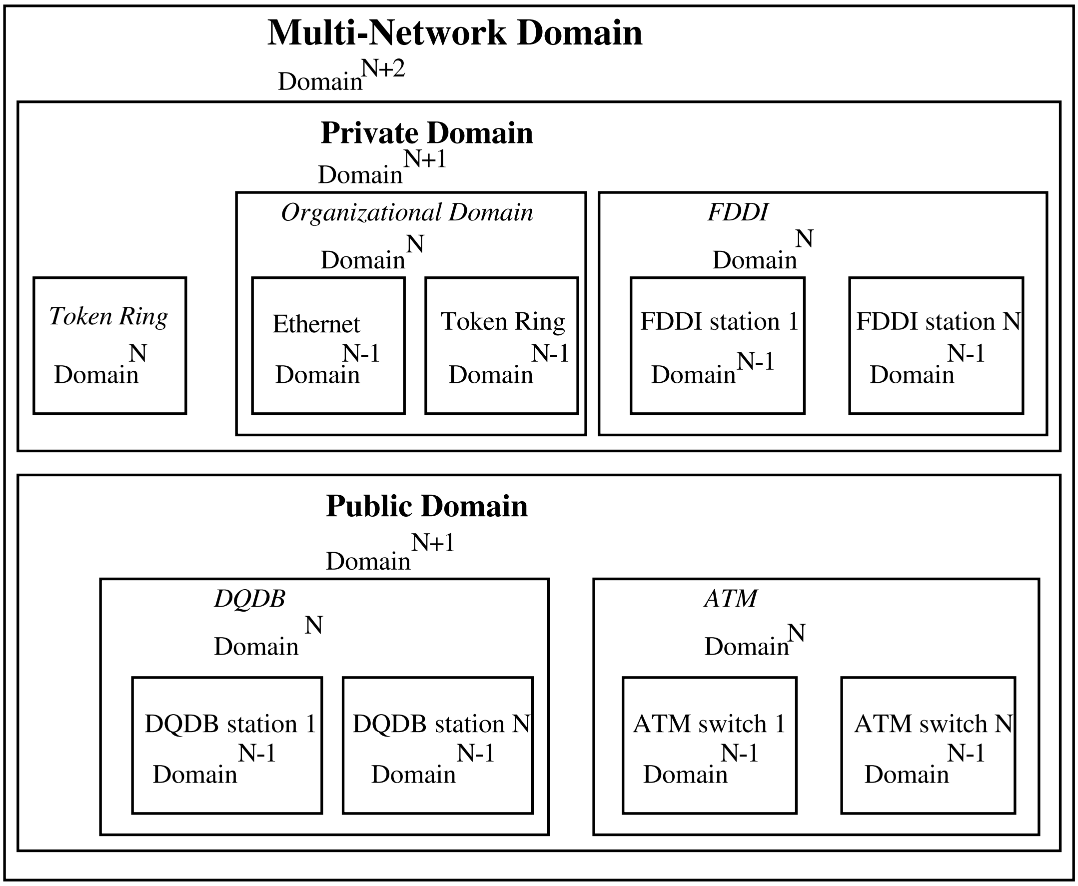



**Figure 7: structuring of the management of the multi-network into domains**

Subdomains have been defined for every station within the FDDI and DQDB domains, and for every switch and connectionless server within the ATM domain. This subdivision is *organizational* since it is restricted by the associated standards. For instance, every FDDI station has its own management part named SMT (Station ManagemenT) [19]. The global FDDI management is realized through cooperation between SMTs. Similarly, in DQDB and ATM subnetworks, the management entities are distributed among stations (DQDB) and switches or connectionless servers (ATM) respectively.

In the remainder of this section, emphasis lies on the structuring of the FDDI Station domain.

## II.3. FDDI Station Domain and its SubDomains

The FDDI Station domain is subdivided into RMT and CMT domains. The applied criterion is the *manager functionality criterion*. The MAC layer manager (RMT: Ring ManagemenT) is responsible of monitoring MAC operation and takes actions necessary to aid in achieving an operational ring. The physical layer manager (CMT: Connection ManagemenT) controls the establishment of a media attachment to the FDDI network, the connections with other nodes in the ring, and the internal configuration of the various entities within a station. It is possible to have more than one MAC in a station. In a station with multiple MACs, there would be one RMT per MAC. Therefore the FDDI Station domain may encapsulate several RMT domains and one CMT domain (see Figure 8).

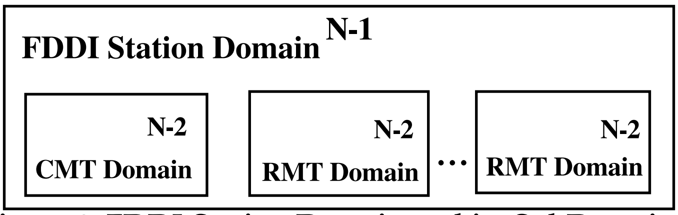



**Figure 8: FDDI Station Domain and its SubDomains**

## II.4. FDDI Management Domain and QoS

After having defined domains for the management of the multi-network architecture, it is required to define management services which also correspond to domains. Configuration, performance, fault, accounting, and security management are examples. *The applied criterion is the manager functionality criterion.* In this area, we focus on Quality of Service (QoS). We model the QoS as a set of criteria.

The considered Qos Criteria are availability, reliability, transit delay, throughput, loss rate, and error rate [20]. For computation of the global QoS (QoS of an FDDI subnetwork, FDDI domain) it is necessary to collect and aggregate QoS of each station on the FDDI subnetwork (FDDI Station domain), themselves an aggregation of those measured at the MAC level (RMT domain). For clarity, we do not consider the CMT domain since QoS criteria measured at this level do not influence the QoS of the Station FDDI domain.

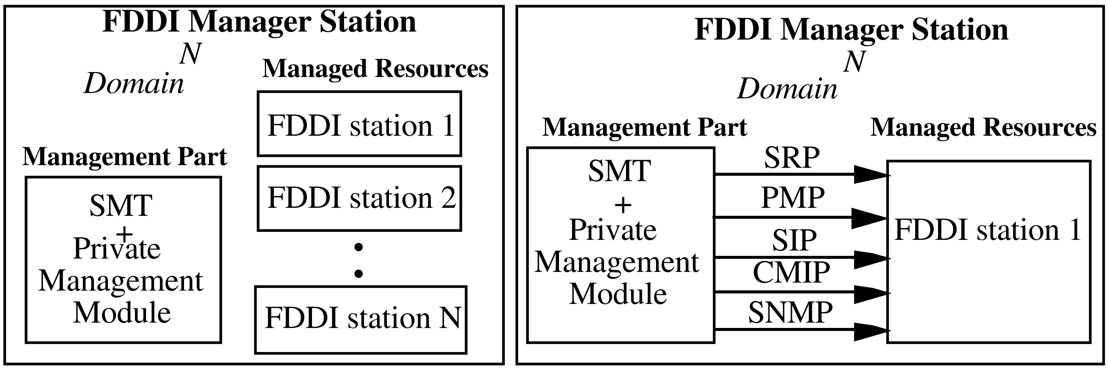
This QoS information is requested through SMT protocols (see Figure 9). These are SRP (Status Report Protocol), PMP (Parameter Management Protocol, and SIP (Status Information Frame Protocol). They enable an SMT entity (the SMT of the FDDI Manager) to request any remote SMT entity (agent for the FDDI manager):

• SRP is performed by any FDDI station to periodically announce its status that is useful in managing an FDDI ring;

• PMP enables access to the FDDI management information base of a remote station. Possible operations are "Get" and "Set". "Get" is used to obtain an attribute value, a group of attributes, or a range of attributes. "Set" enables a management station to remotely configure a station.

• SIP is used to request and provide in response, an FDDI station's configuration and operating information.

Note that use of SNMP [21] or CMIP [22] management protocols is also possible.




**SRP**: Status Report Protocol **PMP**: Parameter Management Protocol **SIP**: Status Information Protocol

**Figure 9: QoS of a global FDDI subnetwork**

In the following, we compute QoS criteria at the MAC and the station levels. From the obtained results, we show how to measure these criteria at the FDDI subnetwork level.
The proposed formulas for QoS computation use FDDI MIB parameters. These parameters should be polled according to a defined polling period. Polling can lead frequently to a significant portion of the network bandwidth being consumed by management traffic. To minimize the resulting number of control frames to be generated by the SMT protocols studied above or by system management protocols such as CMIP or SNMP, we have considered a limited number of MIB parameters within these QoS formulas summarized in Figure 10. These attributes are briefly explained in the following sections.

| Attribute name | Abbreviated registration information according to the SMT MIB |
|---|---|
| PCMState | fddiPORT 62 |
| CurrentPATH | fddiPORT 16 |
| LerEstimate | fddiPORT 51 |
| TraceStatus | fddiPATH 14 |
| RMTState | fddiMAC 111 |
| MA-UnitdataAvailable | fddiMAC 116 |
| FrameErrorRatio | fddiMAC 96 |
| Transmit-Ct | fddiMAC 73 |
| Frame-Ct | fddiMAC 71 |
| Lost-Ct | fddiMAC 82 |
| Error-Ct | fddiMAC 81 |
| NotCopied-Ct | fddiMAC 84 |
| NotCopiedRatio | fddiMAC 105 |
| ECMState | fddiSMT 41 |
| CF-State | fddiSMT 42 |
| PeerWrapFlag | fddiSMT 46 |

**Figure 10: MIB attributes used for QoS computation**

### II.4.1. QoS criteria measured at the FDDI MAC level

The availability criterion is expressed by the boolean "state". It depends on the current state of the RMT state machine (RMTState) and the MA_Unitdata_Enable flag in RMT (MA-UnitdataAvailable). Both can be found in the FDDI MIB [19].

State = ((RMTState == RING_OP) OR (RMTState == RING_OP_DUP) AND MA-UnitdataAvailable).

The *Reliability* criterion is represented by the real variable *Behavior*. The reliability of the MAC network elements (NEs) is computed from the *FrameErrorRatio* and *NotCopiedRatio* attributes of the FDDI MIB. The *FrameErrorRatio* attribute represents the ratio of the packets that have been found with error, for a specific logical path between two neighboring MAC entities. The *NotCopiedRatio* attribute represents the ratio of the correct packets that could not be copied from this MAC (because of a full buffer, for example). Both *FrameErrorRatio* and *NotCopiedRatio* attributes take values in a range from 1 to 65536:

$1 \leq FrameErrorRatio \leq 4 \Rightarrow Behavior1 = 7/7 = 1$
...
$16384 \leq FrameErrorRatio \leq 65536 \Rightarrow Behavior1 = 1/7 = 0{,}14$

$1 \leq NotCopiedRatio \leq 4 \Rightarrow Behavior2 = 7/7 = 1$
...
$16384 \leq NotCopiedRatio \leq 65536 \Rightarrow Behavior2 = 1/7 = 0{,}14$

Finally, $Behavior = Behavior1 \times Behavior2$

The *Throughput* criterion is measured thanks to the MACTransmit-Ct attribute that is encapsulated by the MAC object of the FDDI MIB. It is a count of the number of frames transmitted by the current MAC.

The *Error rate* and *Loss rate* criteria are measured for every MAC NE using FDDI MIB attributes (MACFrame-Ct, MACLost-Ct, MACError-Ct, MACNotCopied-Ct) according to the following formulas:

$$Error\_rate = {}^{MACError-Ct}\!\big/_{MACFrame-Ct}$$

$$Loss\_rate = {}^{(MACLost-Ct+MACNotCopied-Ct)}\!\big/_{MACFrame-Ct}$$

MACFrame-Ct is a count of the number of frames received by the current MAC. MACLost-Ct is a count of the number of instances that this MAC detected a format error during frame reception such that the frame was stripped. MACError-Ct is a count of the number of frames that were detected in error by the current MAC. MACNotCopied-Ct is a count that indicates the number of frames that were addressed to this MAC but were not copied into its receive buffers.

**II.4.2. QoS criteria measured at the FDDI station level**

*Availability* is measured thanks to the two FDDI MIB attributes*ECMState* and *CF-State*. ECMState represents the current state of the ECM state machine. The ECM is in charge of coordinating the trace within an FDDI node. CF-State represents the station's configuration management state. Availability is represented by the boolean "State".
State = (ECMState == IN) AND (CF-State ≠ ISOLATED).

*Reliability*: In a single fault-free FDDI network two logical rings (i.e., two independent data paths) may exist. Therefore the characteristics of these distinct data paths should be considered separately. Consequently, at the Station level, reliability is computed for every logical ring as the product of the MACs that reside in the station and belong to the same logical ring. In order to have the information regarding the current logical ring of every MAC, an attribute (namely *"Ring Map"*) should be added to the FDDI MIB within the MAC object. This attribute may have the form of a connection table and shows the MAC entities that are attached to the same logical ring. Therefore, at the Station level two *Behaviors* are calculated, each for one logical ring.

$$Behavior1(station) = \prod_{\substack{MAC \in Logical\_ring1 \\ MAC \in Station}} Behavior(MAC) \text{ and } Behavior2(station) = \prod_{\substack{MAC \in Logical\_ring2 \\ MAC \in Station}} Behavior(MAC)$$

The station's *Behavior1* and *Behavior2* attributes represent the degradation of the data quality when transiting through the specified station.

*Throughput* is calculated for every logical ring as the sum of the throughput of the MACs that reside in the station and belong to the same logical ring.

$$Throughput1(station) = \sum_{\substack{MAC \in Logical\_ring1 \\ MAC \in Station}} Throughput(MAC)$$

$$Throughput2(station) = \sum_{\substack{MAC \in Logical\_ring2 \\ MAC \in Station}} Throughput(MAC)$$

If the station is a dual attachment station, then

Throughput(station) = Throughput1(station) + Throughput2(station)

*Error rate* and *Loss rate* at the FDDI station level are also considered for every logical ring and are computed from the information provided by the MAC level.
We first consider the following three formulas:

$$ES(R1) = \sum_{\substack{MAC \in Logical\_ring1 \\ MAC \in Station}} MACError-Ct(MAC) \text{ and } ES(R2) = \sum_{\substack{MAC \in Logical\_ring2 \\ MAC \in Station}} MACError-Ct(MAC)$$

$$LS(R1) = \sum_{\substack{MAC \in Logical\_ring1 \\ MAC \in Station}} \left(MACLost-Ct(MAC) + MACNotCopied-Ct(MAC)\right) \text{ and}$$

$$LS(R2) = \sum_{\substack{MAC \in Logical\_ring2 \\ MAC \in Station}} \left(MACLost-Ct(MAC) + MACNotCopied-Ct(MAC)\right)$$

$$RS(R1) = \sum_{\substack{MAC \in Logical\_ring1 \\ MAC \in Station}} MACFrame-Ct(MAC) \text{ and } RS(R2) = \sum_{\substack{MAC \in Logical\_ring2 \\ MAC \in Station}} MACFrame-Ct(MAC)$$

where *RS (Ring i)* represents the number of frames received by the station' MACs placed on Ring i; i may be primary or secondary ring. Note that a Double Attachment Station in FDDI may have one MAC placed on each ring and these MACs may transmit or receive simultaneously data on both rings;
*ES (Ring i)* represents the number of frames received in error from Ring i;
*LS(Ring i)* represents the number of frames received on Ring i and destinated to this station but not copied due to buffer overflow for example.
Then,

$$Error\_rate1(station) = {ES(R1)}\big/{RS(R1)} \text{ and } Error\_rate2(station) = {ES(R2)}\big/{RS(R2)}$$

$$Loss\_rate1(station) = {LS(R1)}\big/{RS(R1)} \text{ and } Loss\_rate2(station) = {LS(R2)}\big/{RS(R2)}$$

### II.4.3. QoS criteria measured at the FDDI subnetwork level

*Availability:* If the number of stations initially configured in the network is *NbStations*, then the *State* of the network is given by:

$$State(Network) = \left( \sum_{Station \in Network} State(Station) \Big/ Nb\_Stations \right) \times State(Ring)$$

The *State(Ring)* attribute represents the contribution of the Ring availability into the global network state. This contribution is necessary because a network without a secondary ring has significantly fewer possibilities of reconfiguration in case of fault on a physical link.
The State of the Ring element represents the availability of the double ring ("trunk ring") of the FDDI network. Obviously, a primary ring is always available. The secondary ring is available when the double ring has not been reconfigured. A reconfiguration is present when at least one station has its PeerWrapFlag set to 1. Therefore,
*if* there is no station with its PeerWrapFlag set to 1 *then* State(Ring) = 1
*else* State(Ring) = 0,5

*Reliability*: Because the FDDI network is a ring and the degradation of any part of the ring affects the whole network, the *Behavior1 and Behavior2* variables (that represent the *Reliability* for every data path) are defined as a product:

$$Behavior1(Network) = \prod_{Station \in Network} Behavior1(Station)$$

$$Behavior2(Network) = \prod_{Station \in Network} Behavior2(Station)$$

*Behavior1(Station)* characterizes the segment of the data path that passes through the station and belongs to one of the two logical rings (namely the *primary ring*), while *Behavior2(Station)* characterizes the segment of the data path that passes through the station and belongs to the other (*secondary*) logical ring. In case of one operational logical ring, only *Behavior1(Network)* is evaluated.

The *Transit Delay* criterion requires a scenario for its measurement since no information is present within the FDDI MIB for that purpose. The FDDI network manager orders the SMT entity of the station where it is located to generate a frame. At reception of this frame, it calculates the time spent by the frame to go around the ring. The occurrence of this operation depends on the polling period. Note that we consider this criterion at the subnetwork level only.

*Throughput*, *Error rate* and *Loss rate* at the FDDI subnetwork level are considered for every logical ring and are computed from the information provided by the FDDI Station level.

$$Throughput1(Network) = \sum_{Station \in Network} Throughput1(Station)$$

$$Throughput2(Network) = \sum_{Station \in Network} Throughput2(Station)$$

$$Error\_rate1(Network) = \left( \sum_{Station \in Network} ES(R1) \right) \Big/ \left( \sum_{Station \in Network} RS(R1) \right)$$

$$Error\_rate2(Network) = \left( \sum_{Station \in Network} ES(R2) \right) \Big/ \left( \sum_{Station \in Network} RS(R2) \right)$$

$$Loss\_rate1(Network) = \left( \sum_{Station \in Network} LS(R1) \right) \Big/ \left( \sum_{Station \in Network} RS(R1) \right)$$

$$Loss\_rate2(Network) = \left( \sum_{Station \in Network} LS(R2) \right) \Big/ \left( \sum_{Station \in Network} RS(R2) \right)$$

## II.5. Relationships between FDDI related domains and QoS

In Figure 11, we represent the FDDI (N), FDDI Station (N-1) and RMT (N-2) domains (For clarity, we omit the CMT domain). RMT controls the placement and operation of a MAC on a ring. The RMT *shield* is concerned with the FDDI information model at the MAC level. This model is described in [23]. The RMT domain is seen as a managed resource from the FDDI Station domain. The RMT provides the FDDI Station (SMT) status information. According to the proposed scenario, this information is related to MAC QoS criteria. With this information, in addition to that provided by the FDDI Station information model [23], the FDDI Station Kernel is able to measure QoS criteria at the station level and activate plans.

The FDDI Domain concerned with the whole FDDI subnetwork encapsulates the FDDI Station domains. The FDDI Shield consists of the FDDI Subnetwork information model described in [23] plus QoS criteria that can be polled from every FDDI Station domain. For these polling operations we have already presented the appropriate protocols, i.e., SRP, PMP, and SIP. Moreover CMIP or SNMP management protocols may be considered. In this latter case, the MIB to be polled is not the one defined in the SMT standard but rather a CMIP conformant FDDI MIB or the SNMP FDDI MIB as defined in the RFC 1512 [24]. Note that the FDDI management standard [19] only focuses on the management of an FDDI station without providing information for the monitoring and control of the global FDDI subnetwork.

At the multi-network level, the hierarchical manager responsible for a private domain (Kernel of the Private domain) requests information concerning the global FDDI network state (QoS criteria measured at the FDDI subnetwork level) from the FDDI manager. Standardization of criteria computed by every network component facilitates cooperation and decision making at any management level.

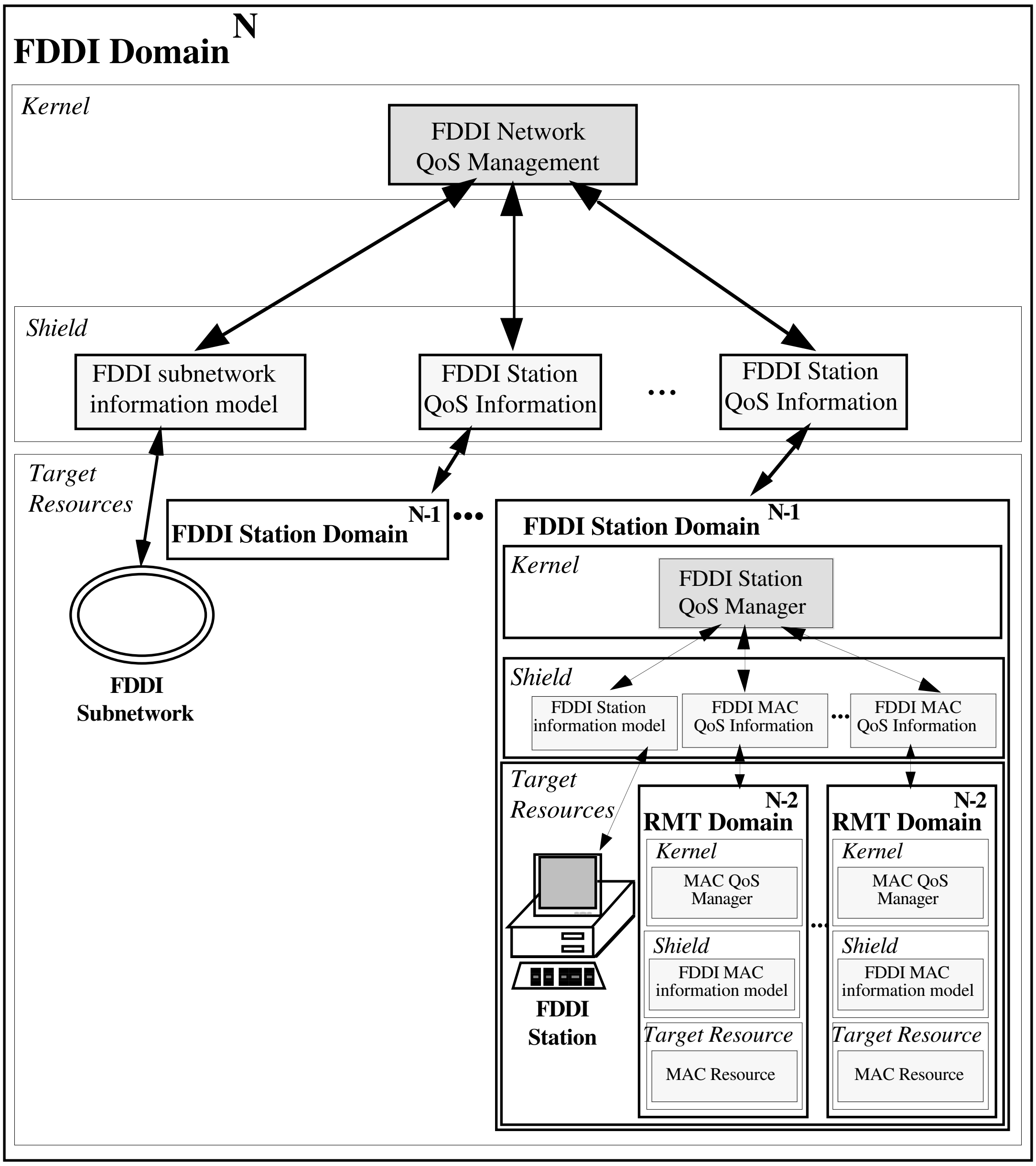



**Figure 11: Relationships between the FDDI (N), FDDI Station (N-1) and RMT (N-2) Domains**

## II.6. Example of QoS management in the FDDI domain

After having computed QoS criteria, emphasis should lie on how to use this QoS information to manage an FDDI subnetwork. In the remainder, we focus on the FDDI domain.

The first step in applying the previous approach to achieve automatic QoS management within the FDDI domain consists in determining the management kernel ingredients that are :
- State attributes, i.e. status of the managed resources in this domain,
- Policy attributes such as thresholds and management options, and,
- Actions.

The second step consists of setting up a plan activation process. A plan is a set of actions to be activated when certain situations occur. A situation is determined by the values of the previous attributes. Typically, a plan activation process is handled as follows:

- Incoming messages (e.g. event notification);
- Update of state attributes;
- Check against policy attributes;
- Invocation of action(s), i.e. the corresponding plan.

QoS Manager State Attributes :

- Error rate (ER);
- Transit delay;
- Availability;
- Throughput [1].

QoS Manager Policy Attributes :

- Information flows automatic rerouting option (ARO);
- Error rate threshold (ERT);
- Availability threshold (AT).

QoS Manager Actions :

- Automatic rerouting of flows;
- Renegotiation of T_req value [2];
- Notification of Fault manager [3];

---

[1] Several flow types with their associated throughput are considered according to their sender and their receiver. These distinctions are required since we operate in a multi-network environment where interworking units play a major role. These flow types are: internal flows that are addressed to a station within the current FDDI subnetwork (IN/IN), external flows that are addressed to a station within the current FDDI subnetwork (OUT/IN), internal flows that are leaving the current FDDI subnetwork (IN/OUT), external flows that are leaving the current FDDI subnetwork (OUT/OUT). Some actions to be performed on a flow depend on the flow type.

[2] During the claim process, each station puts in a request for how fast it wants the token to rotate. This requested rotation time is the T_req. The TTRT is the lowest winning T-req. The TTRT is the only timer that the user can change to optimize data movement.

[3] QoS is a generic network management function that may serve all other network management functional areas [25].

- Deactivation of a station if the LEM function does not react[4];
- Rejection of a particular flow.

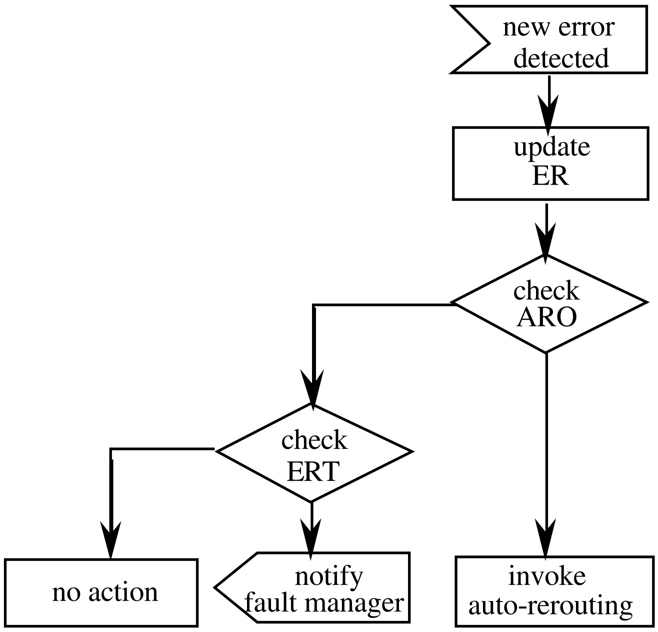



**Figure 12: Management Plan for QoS Management in FDDI Subnetwork Domain**

## Conclusion

Management of large scale networked systems is a complex task due to the number and the diversity of components and activities attached to them. In order to cope with such complexity, we have introduced a domain based management architecture for structuring large scale systems. The architecture allows on the one hand, domains to be completely autonomous in that they manage one or more resources without external interference. Thus a self managed resource can be described easily. On the other hand domains can delegate certain management tasks to other domains allowing the creation of complex management schemes. Real organizational structures and dependencies as well as the inclusion of foreign management systems can be modelled using this approach. Beside handling management systems complexity, the domain concept is also an efficient means for handling the system ability to scale. Dynamic creation/deletion of domains, adding/removing of resources, moving resources from one domain to another and other operations can be easily supported.

Today's challenge is to provide automated support not only for detecting and responding to trivial network and system events but also for the process of planning and policy making to handle more complex situations. However, such automation is hampered by the complexity of the concept of management policies. This is typified by lack of formal notations by which to

---

[4] The errors that can cause most damage are those which occur when the ring is operational. To detect such errors, SMT requires every port to have a link error monitoring (LEM) function. Once the connection is active, a link error monitoring function is implemented for continuous monitoring of the link. The LEM outputs a link error rate based on the errors detected.

capture policy statements and the lack of models and tools for the specification, analysis and refinement of management objectives and policies.

This paper has attempted to explore the issues relating to the automation of some of the more complex aspects of networked systems management. In particular, it has given an outline of our approach to modelling the management activity and then used this modelling as a framework for the formalisation of high level management information. Three kinds of management information have been defined to describe the management task: goals, policies and plans, restricting the freedom of performing management step by step. Management automation is made possible through a refinement of management goals and policies within each domain and, in a top down manner, through the different levels of the domain hierarchy.

The concepts introduced in this paper have been applied to structure the management of a high speed multi-network (ATM, DQDB, FDDI). The FDDI domain and its subdomains have been enhanced. For that subnetwork, we have focused on a particular management function, the QoS. We have shown how to compute QoS criteria and how to automate this QoS management. For the purpose of this demonstrator application, the full complexity of our model has been simplified to specify the goals, policies and plans relevant to the QoS manager. Goals are specified as parameterised invocations of the manager; policies are specified within the manager in the form of policy predicates, associated with policy attributes; and plans are represented in the form of plan templates. The propagation of goals down the management hierarchy is carried out by the proactive management process.

Finally, this paper has pointed out several issues requiring further research to achieve effective automation of management. Further work is needed to develop a language and corresponding tools for specifying goals, policies and plans and tools for automatic derivation of plans from policies and policies from goals as well as tools for detecting conflicts between these information structures. In particular, we intend to explore and determine the impact of our derivation model on peer-to-peer interactions. These latter are mainly negotiations between peer managers for the avoidance (respectively the detection and resolution) of conflicts in case of shared management of resources. Peer negotiations for conflict avoidance should be handled as part of the proactive management process, whereas negotiations for conflict resolution should be handled as part of the reactive management process. Both approaches for handling conflicts should involve all steps of our derivation model, i.e., goals, policies, plans and actions.

**Acknowledgment**
*The Authors gratefully acknowledge the contribution of partners in the DOMAINS project. Participating in this project are Philips Gmbh, AEA Technology, Siemens AG, FIT e.V., Roke Manor Research Ltd., MARI Computer Systems Ltd., APM, Telesystemes, University of Athens, and University of Paris VI .*

## References

[1] M. Möller et al. "DOMAINS Basic Concepts for Distributed Systems Management". Proc. of the fifth RACE TMN Conf. on 1991.

[2] D. Erstin. "Controls for Interorganization Networks". in Gligor 1987, pp 249-261.

[3] D. M. Nesset. "The inter-Authentication-Domain (IAD) Logon Protocol, (Preliminary Specification and Implementation Guide)". Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550, 6 May 1988.

[4] D.A. Gomberg. "A Model of Inter-Administration Network User, Authentication & Access Control". Mitre Corporation, Washinton C3I Division, 7525 Colshire, Drive, McLean VA 22102, MTR-87W00003, Dec 1987.

[5] M. Sloman and J. Mofett. "Domain Management for Distributed Systems". Integrated Network Management, vol. 1, B. Meandzija and J. Westcott (Eds.), North Holland, Amsterdam 1989.

[6] B. Wang et al. "Database/domain approach to distributed systems management". Computer Comm. 1989.

[7] Information Proc. Syst, OSI Systems Management Overview, ISO/IEC DIS 10040, 1991.

[8] R. Boutaba, A. Benkiran. "A Framework for Distributed Systems Management". in Proc. of Networks' 92, Int. Conf. on Computer Networks, Architecture and Application, Trivandrum, October 1992.

[9] R. Boutaba. "A Methodology for Structuring Management of Networked Systems". IFIP Conf. on AIP Techniques for LAN and MAN Management. Versailles, April 1993.

[10] Y. Yemini et al. "Network Management by Delegation". in Integrated Network Management II, I. Krishnan and W. Zimmer, Eds. North Holland, 1991, pp. 95-107.

[11] D. Holden. "Predictive Languages for Management". Integrated Network Management, Vol 1, B. Mengzija and J. Wescott, Eds. North Holland 1989.

[12] O. Wolfson, S. Sengupta, Y. Yemini. "Managing Communication Networks by Monitoring Databases". IEEE Transactions on Software Engineering 17, 1991, pp. 944-953.

[13] J.D. Mofett et al. "Specifying Discretionary Access Control Policy for Distributed Systems". Comput. Commun., Vol. 13, no. 9, Nov. 1990. pp. 571-580.

[14] J.D. Mofett, and M.S. Sloman. "The Representation of Policies as System Objects". Proc. Conf. Organiz. Comput. Syst., Atlanta, GA, Nov. 1991.

[15] J. Bedford-Roberts. "Concepts from Pythagoras". Report HPL-91-22, Hewlett Packard Laboratories, Bristol, UK, Feb. 1991.

[16] C.E. Alchourron. "Philosophical Foundations of Deontic Logic and its Practical Applications in Computational Contexts". in Workshop on Deontic Logic in Computer Science, Amsterdam, Dec. 1991.

[17] R. Wieringa et al. "Specifying Dynamic and Deontic Integrity Constraints". in Data and Knowledge Engineering. Vol. 4, North Holland. Amsterdam 1989.

[18] J.D. Mofett, and M.S. Sloman. "Policy Hierarchies for Distributed Systems Management". IEEE Journal on Selected Areas in Communications. Vol. 11. no. 9. Dec. 1993.

[19] G. Milligan, F. Ross. "FDDI Station Management (SMT)". Draft proposal, June 92.

[20] N. Simoni, S. Znaty. "QoS: From Definition to Management". 4th IFIP Conference on High Performance Networking, Dec 92.

[21] ISO 9596. Information Processing Systems - Open Systems Interconnection - Common Management Information Protocol Specification, Jan 90.

[22] J. Case et al. "A Simple Network Management Protocol". RFC 1067, Aug 1988.

[23] J. Sclavos et al. "Information Model: From Abstraction to Application", IEEE NOMS'94, Feb 94.

[24] J. Case. "FDDI Management Information Base", RFC 1512, Sept 93.

[25] R. Boutaba, S. Znaty. "Towards Integrated Network Management: a Domain/Policy Approach and its Application to a High Speed Multi-Network". IEEE NOMS'94, Feb 94.