# Applying the Policy Concept to the Management of ATM Networks

Raouf Boutaba[1] and Ahmed Mehaoua[2]

[1] Telecommunication and Distributed Systems Division,
Computer Research Institute of Montreal,
1801, Av. McGill College, Suite 800,
Montreal (Qc), H3H 2P2 - CANADA -
Email: rboutaba@crim.ca

[2] Laboratoire PRiSM,
Université de Versailles,
45 Av. des Etats-Unis, 78000 Versailles, FRANCE
Email: Ahmed.Mehaoua@prism.uvsq.fr

## Abstract

*The introduction of sophisticated processing capabilities within current networks allows to provide more advanced services to users and a high level of customization. This has led to the need of an integrated management of network, system and application resources. Furthermore, the large number and diversity of the resources to be managed in such processing and communication environments stressed the need for an automated management support to deal with the increasing complexity. This integrated and automated management is based largely upon high level architectural concepts such as management policies. This paper discusses the application of such architectural aspects for the automated management of very high speed networks. The goal is to reduce the control and management delays which are very critical in such environments. In particular, a hierarchical domain structure a recursive policy-based activity are proposed for the real-time management of an ATM switch which is the basic network element of B-ISDN.*

**Keywords:** ATM, network management, automation, policy, reactive control

## 1 Introduction

Asynchronous Transfer Mode (ATM) has been adopted by ITU-T (formerly CCITT) as the transport and switching technique for the Broadband Integrated Services Digital Network (B-ISDN). One reason industry has turned its attention to the management of ATM-based Broadband networks is to ensure that operations impacts and needs are identified and addressed early in the B-ISDN development process. Without the appropriate tools to anticipate, detect, and overcome failures, congestion and periods of degraded performance that may occur within B-ISDN, the viability of this broadband network will be jeopardy.

Usually, ATM network management and ATM traffic control are separated. Indeed, traffic controls are performed automatically by the set of real time algorithms (with response time of the order of microseconds) while management tasks are inherently slow and may often involve (if automated network management tools are not provided) a human manager [1]. However, network management and traffic control operations interact each other. Indeed, the two basic traffic management and control functions (CAC and UPC) are necessary to guarantee the desired broadband network performance [2], [3]. Therefore, an automated management is being increasingly demanded [4].

This paper addresses management automation by refining a uniform and recursive policy-based management model and its application to end-to-end ATM connections. It is structured in two main parts. In the first one, we introduce a policy-based structuring for organizing management systems and a policy-driven management activity. These concepts are based on results of the DOMAINS ([5], [6]) project. The model allows for recursive and generic structuring we consider as the basis for management automation. It introduces management policies as statements on how a manager will achieve his/her management objectives by refining them into executable management plans.

In the second part, we apply policy-driven approach to the management of an ATM-based network highlighting VCC (Virtual Channel Connection) and VPC (Virtual Path Connection) management structuring. As an example, an automated policy-based control activation process for an ATM switch management is presented.

## 2 The Policy Concept

The major effort handled within standardization activities has been on the modeling of management aspects related to OSI communication systems. More recently, some standardization activities has been allocated to deal with management applications covering all aspects of ODP systems. It is the case of the Working Group 4 of Sub-committee 21 (ISO/IEC JTC1/SC21/WG4) who is collaborating with ITU for the development of systems management recommendations documented in the X.700 series [7]. However, there is still a lot of work to do in order to achieve a fully integrated management of networks and distributed systems. In particular, current developments fail to achieve a uniform management of all system resources involved into an organization which may be physical, logical, communication and/or processing resources. They haven't sufficiently addressed the roles and activities which exist within such a system's organization, the interactions between the system and the environment in which it is placed, the way the system is organized, and the policies which apply. Management policies are insufficiently defined and their usage and implementation are still open issues.

Still humans are in charge of interpreting and applying high level policies which are often abstract, informal and vary from a system to another according to organizational, social, human or technological factors. The complexity of the tasks of making and interpreting (informal) policies hampers the automation of the management process. Several undergoing research works are addressing the subject of policies specification and their application to the management of networks and distributed systems. Among the more significant ones are those held in the scope of the Domino Project [8], but they have mainly focused on security policies implemented as access rights [9]. 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 [10]. This way a policy can be defined as a statement on how a manager will control the resources under her/his responsibility in order to achieve her/his goals. Similar approaches favoring such a distinction are focusing on the provision of support tools and services for the definition, manipulation and enforcement of management policies. In [11] a platform is described which includes a set of services for policy management (i.e., creation, deletion, modification, etc.) and policy enforcement and refinement (interpretation and mapping to low level policies and infrastructure-specific procedures). In ([12], [13]) classifications of management policies are highlighted as a means to aid in the policy specification and transformation process. The resulting policy classes hierarchy consists of policies at the lower levels that can be formalized and hence automatically interpreted, but more abstract policies at the higher levels.

For the specification of abstract policies, few works have identified deontic logic, the logic of normative systems, as good candidates ([14], [15]). Indeed, this logic 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. ([16]) 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.

On the whole, it can be said that there is an increasing need for a precise definition of policy and its role within the management process as well as a good comprehension of the way in which policy directs the behavior of managers in controlling the managed system resources. There is also a need for support tools for the definition, manipulation and enforcement of policies. These are all prerequisites for the provision of an automated policy-driven management activity.

## 2.1Policy-driven Management

Policy in standards and in the major research works is being referenced as the overall rules governing the management of all members of the same domain. A domain is introduced here as the collection of managed resources to which the policy applies, i.e., the sphere of influence of the manager implementing the policy. But the domain concept has been more largely used as an efficient means to structure a management system into management subsystems in order to reduce the complexity of the overall management task, adhering this way to the "divide and conquer" philosophy. Different criteria can be used for the construction of management domains that can be related to geographical, organizational, security, functional structuring, etc. [17]. The applied policy is one of these criteria which allows to built domains applying specific management policies, but policy-based structuring may overlap with other structuring schemes. These considerations highlight the need for a good structuring of the management system and a good organization of the overall management activities which constitute the base of a good quality management. The structuring task is more likely ensured by a management system designer who has a good knowledge of the managed system features and behavior on one hand, and a sufficient experience on the design of managing systems on the other hand.

In order to help designers in this task, we have proposed a management architecture which can be used as a guideline for building and organizing management systems [17]. Two structuring principles underlie this architecture. The first is the use of domains as a means for grouping resources determined by management needs. The dependency between the emerging domains reflects both hierarchical interactions (e.g., control of resources, authority delegation to subordinate managers [18]) 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, to provide managers with an abstract and uniform view of the managed resources hiding these last heterogeneity and, on the other hand, to implement a uniform and generic model for the management activity. We believe that defining and applying such a uniform and recursive management model is the key step towards management automation.

## 2.2 Policy-based Management Model

Most of existing management systems implement the standard manager-agent architectural model. It is clear that the roles of the agents and the managers are affected by the applied management policies. However, it is the way these management entities are affected by policies that needs to be refined in order to better implement policy-driven management activities. The difficulty comes from the fact that 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 which hinders integration and open cooperation.

It is our goal to refine the management model by introducing policies as the means for driving the management activity. In this perspective, we define three kinds of management information: goals, policies and plans. The manager 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 (control actions) on the occurrence of certain situations (observed state). The manager reports about its management to its superiors particularly when it fails to achieve the goals. Management policies are introduced as intermediaries between (possibly abstract) management goals and (often executable) management plans. They are general statements about how management goals will be achieved, and are used to ease decision making .

According to this model, the activity of the manager consists of (1) making management policies from management goals; (2) deriving management plans from management policies; (3) executing management plans. Management plans are conditional sequences of executable control actions. They are executed to react on events detection or arrival of notifications from managed resources. Adhering to this model enables the dynamic change of goals, policies and plans which are currently very often hard-coded into management applications.

In the scope of a hierarchical management system, the plans of a given manager can be equated to the goals assigned by this manager to its subordinates. This way management policies are recursively through the different levels of the management hierarchy. At the highest levels, management goals, policies and plans may be abstract. In this case, the policies help in the process of coherently subdividing goals into subgoals and of distributing this subgoals to appropriate subordinate managers. It is also the role of the policy to prevent the occurrence of conflicts between subgoals. At the leaves of the management hierarchy, goals, policies and plans are more concrete as they take into account the real state and behavior of the managed resources in addition to the high level objectives. In this case, the policies determine which management plans to be executed on the managed resources and help in the process of detecting potential conflicting control actions and the resolution of these conflicts.

This policy-based model based on the derivation of management policies from goals and their transformation into control plans is performed by all managers in the management system. Managers interact in a hierarchical fashion in the scope of the domain hierarchy. Each domain includes a manager which receives goals from superiors and reports results to these superiors. The manager also sends subgoals to subordinates and receives their reports. Managers also interact in a peer-to-peer fashion if a cooperation is necessary to achieve a global objective when sharing the same resources for example. Policies can also be used to encourage and regulate such cooperation or to resolve conflicts if they occur.

## 2.3 Policy Representation

The implementation of policy-driven management activities is being hampered by the lack of tools for representing policies formally. The complexity of the tasks of making and interpreting (informal) management policies make difficult such automation. 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. Currently, the automation of policy-driven management is ensured only at a very low level since policies are closer to the specifics of the real resources and can thus be expressed in an algorithmic way. At upper levels policies are more likely similar high level principles expressed in a natural language. In this case only humans are able to perform the transformations unless these high level policies are represented in a formal way.

Towards this formalization and hence automation of policy driven management, we adopt a predicate-based representation to express policies and algorithmic sequences of actions to define management plans. Starting, for a given manager, with a well defined set of goals, 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 *Plan Template*. This assumes a good knowledge of the resources to be managed as well as the control actions to be performed on these resources. A plan template consists of a tree of action sequences with state and policy decisions at each branch. Policy predicates are represented as explicit calculations to determine a policy decision.

Based on this representation choices, the first step to apply our policy-driven management approach consists of determining the manager ingredients that are:

- Goals,
- 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.

In the sequel, we will apply this policy-driven model to automation of the management of an ATM switch where the time constraint is particularly critical.

# 3. ATM Policy-driven Management

In this section, we will first define the management architecture for the ATM network. This consists to identify the management domains based on the resources to be managed in the ATM environment. Secondly, we will focus on the ATM switch domain in order to show the feasibility of our policy-driven management approach. The formal specification of the ingredients intervening in the policy-based model will allow us to provide management plans that can be automatically activated to react to the occurrence in the system of certain events. The main goal of management automation in the ATM context is to be able to efficiently react to the occurrence of significant situations and carry out the necessary real-time control operations.

## 3.1 ATM Management Architecture

In an ATM network, due to the large number of VP connections and VC connections that may be configured it is not practical to *manage* every one of them, although they will be systematically *controlled.* VCCs and VPCs that have relatively long or permanent holding times are mainly those that are candidate for management and are those for which domains are created. These last are encapsulated in the highest level ATM domain responsible of all end-to-end connections management. Each VCC domain is responsible of an end-to-end connection management and thus of the corresponding lower level connections (VPCs). Therefore, a domain is created for each VPC involved in the VCC connection. The management entities of a VPC domain are distributed among ATM switches which leads us to consider each VPC domain as a set of ATM switch domains. The management of each switch at the frontier between two VPC (i.e. where new VC numbers are assigned [9], [10], cf. figure 1) is shared by these VPC domains.

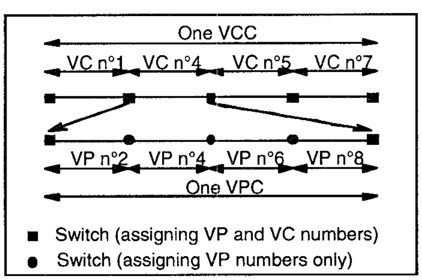


**Figure 1 : ATM end-to-end connection**

Every ATM Switch which is the basic network element of B-ISDN is composed of ports. Every port supports many connections. A port identifies a virtual path (VP), a connection identifies a virtual channel (VC). Note that a VP is a generic term for a bundle of virtual channel links while a VC is a generic term used to describe a unidirectional communication capability for the transport of ATM cells. These three management domain levels, namely Switch, Port, and Connection do not share the same management functionality, i.e. the management is performed differently on each of these domains (see figure 2)

In the following sub-section, we present, as an example, the switch domain manager through a reactive approach. Due to its position in the management hierarchy, the switch manager receives its management goals from higher level managers (VCCs through VPCs) and delegates management sub-tasks to lower level managers (namely VP and thus VC managers).

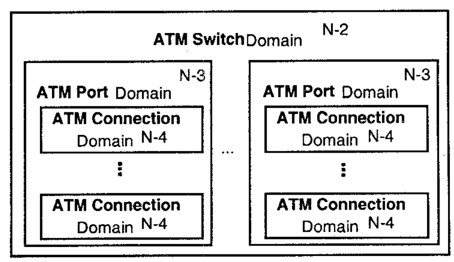


**Figure 2 : Domain-based structuring of an ATM Switch**

## 3.2 Switch Domain Manager

In the scope of example application, we have designed the switch manager in such a way that only a limited and well defined set of goals is specified, from which a set of policies are determined automatically and then turned into a plan of actions by applying defined policy predicates to the predetermined set of plans.

We have defined 3 high level management goals related to ATM end-to-end connections performance management. These goals are the following :

- Goal G1 :
  Loss rate for data flow is limited to p% of transmitted cells on VCC n°i with priority P1.
- Goal G2 :

Transit delay for voice and video flows is limited to T for VCC n°i with priority P2.

- Goal G3 :
  Control and management delay, with priority P3, is limited to p% by unit of time; beyond this limit, all actions with priority less than P3 are forbidden.

Among the more significant state attributes to ATM performance management, there are ([19], [20]):

- Cell Loss Ratio (S-CLR) ,
- Cell Misinsertion Rate (S-CMRe) ,
- Cell Misinsertion Ratio (S-CMRo),
- Cell Error Ratio (S-CER),
- Cell Block Error Ratio (S-CBER),
- Severely Errored Cell Block Ratio (S-SECBR),
- Cell Transfer Delay (S-CTD) ,
- Mean OAM CTD (S-MOC),
- OAM Cell Delay Variation (S-OCDV).

We add the following ones :

- S-avai : a Boolean reflecting the availability of the switch;
- VP-i-avai : a set of Boolean reflecting the availability or not of each VP managed in the switch;

The policy attributes that are relevant to cell loss and transit delay are :

- S-CLR-max : maximum cell loss rate accepted for the traffic;
- VLB : use Virtual leaky bucket algorithm option;
- Discard : discard (or not) cells with CLP bit set to 1 option;
- EBCI : use explicit backward congestion indicator option;

Now, from these policy and state attributes and in conjunction with the set of goals, we can decide the composition of policies. Two policy predicates can be defined for the switch manager :

- Policy predicate P1 :
  **if** S-CLR > S-CLR-max
  **then** **if** [ S-CLR > S-CLR-max *1.01
  **&** P(Loss rate) < P(Ctrl delay) ]
  **then** deactivate VP (-->"no")
  **else** activate virtual leaky bucket
  (-->"yes")
  **else** do nothing

- Policy predicate P2 :
  **if** [ P(Transit Delay)> P(Loss rate)
  **&** S-CLR < S-CLR-max ]
  **then** discard cells with CLP set to 1
  (-->"yes")
  **else** send OAM cells with EBCI mechanism
  (-->"no")

We have identified which control actions may be performed by a switch manager. These are (but not limited to) :

- A0/A1 : save (respectively ignore) message;
- A2/A3 : set to true (resp. to false) VP-i-avai;
- A4/A5 : execute (resp. stop) virtual leaky bucket;
- A6/A7 : discard (resp. stop discarding) cells with CLP bit set to 1;
- A8/A9 : set to true (resp. to false) VLB attribute;
- A10/A11 : set to true (resp. to false) Discard attribute;
- A12 : send OAM cells with EBCI mechanism;
- A13 : send OAM cells with EFCI mechanism;
- A14/A15 : set to true (resp. to false) S-avai;

According to this, we obtain for the switch manager, the reactive plan activation process described below in the form of state machine, where :

- N1 : notification indicating to the switch manager the values of certain VP-state attributes (e.g. CLR, ...);
- N2 : notification from the switch manager asking the VPC domain manager a connection reconfiguration;
- N3 : notification from the switch manager to the VCC domain (through the VPC domain) concerning bandwidth renegociation;
- S-G1 : this sub-goal is sent by the switch manager (to the VP domain manager) to put the VP out of service.

According to this plan activation process, the switch domain manager receives a notification (N1) from its VP sub-domain which indicates the values of certain parameters such as the cell loss rate attribute (CLR). Within the OAM cells, a parameter reflects the state of the switch with respect to the management activity. If the switch is active, it checks the EBCI (Explicit Backward Congestion Indicator) mechanism. If this last is unavailable, policy predicate

(P1) is computed. Two cases are then possible : - the VP is put out of service if the cell loss rate is not acceptable and a reconfiguration of the connection request is sent to the superior VPC domain; - if the result of predicate P1 is positive, the VLB (Virtual Leaky Bucket) is activated and the superior VC domain is asked for bandwidth renegociation. If the EBCI mechanism was available at the level of the switch, policy predicate (P2) is computed. If a positive result is obtained, next arriving cells marked with bit CLP (i.e., CLP = 1) are discarded; otherwise, all switches down to the source of the cells are notified.

***Switch Manager state-machine***

```
If (Notification N1 arrived)
Then Save message N1;

If Not (Switch)      /* Switch Management not available
Then Put S-avai:=false
Else {Put S-avai:=true;
        If Not (EBCI)    /* EBCN function not available
        Then if (P1)
               Then {/* Low congestion level
                        Put VLB_attribute:=true;
                        Execute VLB;
                        Send Notification N3 for
                        bandwidth renegociation
                        }
                Else  {/* High congestion level
                        /* VP-i is set out-of-service
                        Send sub-goal S-G1
                        Put VP-i-avai:=false;
                        Send Notification N2 for
                        connection reconfiguration
                        };
        Else if (P2)
              Then  {
                        /* Delay-sensitive
                        Put Discard_attribute:=true;
                        While (CLP_bit=1)
                        do Cell Discarding;
                        }
               Else  {/* Loss-sensitive
                        Send OAM cells with EFCI;
                        }
       }
```

# 4 Conclusion

In this paper, we have introduced a policy-based model and applied it for the real time management of an ATM switch. Automated management support tools are increasingly demanded to support real-time management activities and to free human operators from the complex task of managing current very large networking and processing environments. We have also shown how the introduced management model can be recursively refined in the scope of a management hierarchy. The model is based on a policy-driven activation of management plans and their execution to respond to high level goals.

In this paper, emphasis lies on the switch domain manager, this being a basic network element in B-ISDN. In particular, we have presented a reactive plan for automated performance management at the level of the switch domain. This plan is more adapted to the transport of data which is sensitive to cell loss. For voice and video transport which are highly dependent on transit delays, proactive policies will be more helpful due to its ability to anticipate problems that may occur in the network and, particularly, periods of performance degradation. Proactive plans for ATM switch management, the corresponding policy predicates together with the overall end-to-end view of ATM management are under development.

For the purpose of the demonstration, we have simplified our model and applied it to the management of the ATM switch. Although this has shown the viability of the management model and the feasibility of an automation, it is still necessary to provide a formalism for a complete specification of management policy hierarchies. This will allow to specify low level management policies such as those presented in the ATM example as well as high level ones, e.g. those applying at the level of the business enterprise employing the ATM network and which reflects the management goals of this one.

Few works are exploring the problem of specifying management policies (some have been referenced in the remaining of the paper). But, still a lot of work to do in order to have high-level languages and intelligent tools to define, express, manipulate, analyze, apply management policies and maintain system consistency by preventing and/or resolving potential conflicts.

We are actually investigating the use of deontic logic as a means for formally specifying management policies. The problem is that the standard deontic logic can lead to conflicts in the policy hierarchy such as the 'contrary-to-duty imperative' paradox [21] where two derived obligations are contradictory or the conflict between permission and obligation policies. In order to deal with such conflicting situations, we are exploring the application of defeasible reasoning, a Horn clausal form of logic programming, to deontic logic ([22]) which provides computational means to resolve deontic paradoxes (e.g., the 'incompatible' operator, the ability to set up priorities in policy hierarchies and deontic expressions, etc.).

## 5 References


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

2. ITU-I.371- Traffic Control and Congestion Control in B-ISDN, March 93.

3. N. Yamanaka, Y. Sato, K. Sato. "Usage Parameter Control and Bandwidth Allocation Methods Considering Cell Delay Variation in ATM Networks". IEICE Trans. Comm., Vol. E76-B, N. 3, March 93, pp 270-279.

4. R. Boutaba, S. Znaty. "Integrated Network Management : From Concepts to Application to ATM-Based Networks". IEEE GlobeCom'94, Nov. 94.

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

6. 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.

7. ITU Recommendation X.701 - ISO/IEC/10040 Recommenedation, "Systems Management Overview", 1991.

8. M. Sloman, J. Moffet, K. Twiddle. "Domino Domains and Policies: An introduction to the project results". Domino Report Arch/IC/4, Feb. 92.

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

10. R. Boutaba, S. Znaty. "An architectural Approach for Integrated Network and Systems Management", ACM Computer Communication Review, Oct. 1995, vol 25, n 5, pp. 13-38.

11 B. Alpers and H. Plansky. "Concepts and Application of Policy-Based Management". In Proc. of 4th IFIP Conf. on Integrated Network Management (A. D. Sethi, Y. Renaud and F. Faure-Vincent eds.), 1995.

12. S. B. Calo, M. J. Masullo. "Policy Management: An Architecture and Approach". In Proc. of IEEE Workshop on Systems Management, Cal., Apr. 1993.

13. R. Wies. "Using a Classification of Management Policies for Policy Specification and Policy Transformation". In Proc. of 4th IFIP Conf. on Integrated Network Management (A. D. Sethi, Y. Renaud and F. Faure-Vincent eds.), 1995.

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

15 A. Langsford, J. Moffet. "Distributed Systems Management". Addison Wesley, 1993.

16. 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.

17. R. Boutaba. "A Methodology for Structuring Management of networked Systems". IFIP Conf. on AIP Techniques for LAN and MAN Management. Versailles, April 1993.

18. 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.

19. S. C. Farkouh, "Managing ATM-Based Broadband Networks", IEEE Communications Magazine, may 1993.

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

21. 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.

22. Y. U. Ryu, R. M. Lee. "Defeasible Deontic Reasoning: A Logic Programming Model". Deontic Logic in Computer Science: Normative System Specification, John Wiley & Sons Ltd , 1993.