Temporal-Oriented Policy-Driven Network Management

Manuela-Ionelia Dini
School of Computer Science
McGill University, Montreal
July, 2000

A Thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of
Master in Computer Science

0-612-70702-4

Abstract

Systems management represents the set of activities necessary to ensure that information systems function according to user requirements and objectives. Chapter 1 summarizes the management challenges in today's networks and distributed systems. Policy-driven network management is the new management paradigm. Its implementation needs a new information and decision model, appropriate protocols and new hosting and access mechanisms. IETF policy framework and architecture create the support for the deployment of this new paradigm.

Consideration of temporal aspects to allow run-time policy conflict detection and error-handling has not yet been developed. After assessing the existing policy-oriented achievements in terms of tools and IETF drafts, and after presenting existing temporal mechanisms, we concluded that only policy definition temporal issues are partially referred to. We considered complementary temporal issues focusing on "policy execution" and coping with the notions of time zones, policy dependency across multiple time zones and actions translation among many time zones. Finally, we showed how our input can be used for extending the current IETF Policy CIM proposal.

We intend to have an IETF draft on these issues. Specifically, our proposal can be added as a new building block to any existing policy-enabled management tool. We identified important directions in handling policy conflicts at run-time.

Résumé

La gestion des réseaux représente l'ensemble d'activités nécessaires pour s'assurer que les systèmes d'information fonctionnent selon les exigences et les objectives de l'utilisateur. Le premier chapitre résume les défis de la gestion des réseaux d'aujourd'hui et des systèmes répartis. La gestion des réseaux basée sur des politiques est le nouveau paradigme de gestion. Son implémentation a besoin d'un nouveau modèle d'information et de décision, de protocoles appropriés et de mécanismes d'accès. Le cadre et l'architecture des politiques d'IETF créent le soutien nécessaire pour le déploiement de ce nouveau paradigme.

La considération des aspects temporels pour permettre la détection des conflits entre politiques au moment de l'exécution et le traitement des erreurs n'a pas encore été développé. Après avoir évalué des accomplissements existants orientés vers les politiques en termes d'outils et des propositions d'IETF, et après avoir présenté les mécanismes temporels existants, nous avons conclu que des issues temporelles de définition de politique sont seulement partiellement mentionnées. Nous avons considéré les aspects temporelles complémentaires se concentrant sur l'exécution des politiques et tenant compte de notions de fuseaux horaires, de la dépendance des politiques à travers des fuseaux horaires multiples et de l'équivalence des actions à travers plusieurs fuseaux horaires. En conclusion, nous avons montré comment notre contribution peut être utilisée pour étendre la proposition actuelle d'IETF concernant les politiques.

Nous avons l'intention de proposer des améliorations à la proposition actuelle d'IETF concernant les politiques. Spécifiquement, notre proposition peut être ajoutée comme nouveau module à n'importe quel outil de gestion existant basé sur les politiques. Nous avons identifié des directions importantes a suivre pour traiter les conflits entre politiques au moment de l'exécution.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Prof. Gerald Ratzer, for his continuous guidance and encouragement throughout the course of this work. He helped me shape the ideas and concepts presented here and provided me with the opportunity to discover and develop many interests. His assistance is reflected not only in the contents of this thesis but also in the presentation style.

I would also like to thank the Computer Research Institute of Montreal (CRIM) for providing the necessary technical environment for the completion of this thesis.

I am very thankful to my brother, Cosmin, for his technical help, constructive criticism, and for always being there when I needed a second opinion.

Finally, I would like to thank my entire family for their invaluable support, especially my husband, Sebastian, whose encouragement and understanding helped me fulfill my goal.

List of Figures

List of Tables

Table of Contents

List of Acronyms

AF: Assured Forwarding

API: Application Programming Interface

ASN. : Abstract Syntax Number 1

ATM: Asynchronous Transfer Mode

ATM MIB: Asynchronous Transfer Mode Management Information Base

CBR: Constant Bit Rate

CIM: Core Information Model

CLI: Command Line Interface

CoS: Class of Service

CPU: Central Processing Unit

CMIP: Common Management Information Protocol

CMIS: Common Management Information System

COPS: Common Open Policy Service

CORBA: Common Object Request Broker Architecture

DEN: Directory Enabled Networks

DiffServ: Differentiated Services

DLCS: Dynamic Link Configuration System

DMTF: Distributed Management Task Force

DSCP: Differentiated Services Code Point

EF: Expedited Forwarding

EIGRP: Enhanced Interior Gateway Routing Protocol

EMA: Enterprise Management Architecture

FDDI: Fiber Distributed-Data Interface

FIFO: First-In-First-Out

GDMO: Guidelines for Definition of Managed Objects

GRM: General Relationship Model

GUI: Graphical User Interface

HMMP: HyperMedia Management Schema

HMQL: HyperMedia Query Language

HTTP: Hypertext Transfer Protocol

IANA: Internet Assigned Numbers Authority

IETF: Internet Engineering Task Force

IntServ: Integrated Services

IP: Internet Protocol

IP DA: Internet Protocol Destination Address

IP SA: Internet Protocol Source Address

IPSec: Internet Security Protocol

IPX: Internetwork Package Exchange

ISO: International Standardization Organization

ISP: Internet Service Provider

IT: Internet Technology

LAN: Large Area Network

(L)DAP: Lightweight Directory Access Protocol

LPDP: Local Policy Decision Point

L2TP: Level Two Tunneling Protocol

MAC: Media Access Control

MIB: Management Information Base

MO: Managed Object

MOF: Managed Object Format

MPD: Manager Position Domain

MPLS: Multi-Protocol Label switching

NAT: Network Address Translation

NMP: Network Management Protocol

OMA: Open Management Architecture

OMG: Object Management Group

OSF/DME: Open Software Foundation / Distributed Management Environment

OSI: Open System Interaction

OSPF: Open Shortest Path First

QPM: QoS Policy Manager 1.1 (Cisco Systems)

QoS: Quality of Service

PEP: Policy Enforcement Point

PCIM: Policy Core Information Model

PDC: Primary Domain Controller

PDP: Policy Definition Point

PDU: Protocol Data Unit

PHB: Per Hop Behavior

PIB: Policy Information Base

PPP: Point-to-Point Protocol

RESV: Reservation Request (in the context of RSVP)

RSpec: Request Specification (in the context of RSVP)

RSVP: Resource Reservation Protocol

RMI: Remote Method Invocation

RMON MIB: Remote Monitoring Management Information Base

SBM: Subnet Bandwidth Management

SDH: Synchronous Digital Hierarchy

SLA: Service Level Agreement

SMI: Structure of Management Information

SNA: Systems Network Architecture

SNMP: Simple Network Management Protocol

TACACS+: Terminal Access Controller Access Control System Plus

TCP: Transmission Control Protocol

TL1: Transaction Language One

TME: Tivoli's Management Environment

ToS: Type of Service

TSpec: Traffic Specification (in the context of RSVP)

UDP: User Datagram Protocol

UNMA: Unified Network Management Architecture

URD: User Representation Domain

URT: User Registration Tool

UTC: Universal Time Clock

VLAN: Virtual Local Area Network

VOIP:	Voice over IP (Internet Protocol)
VPN:	Virtual Private Network
WAN:	Wide Area Network
WBEM:	Web-Based Enterprise Management
WFQ:	Weighted Fair Queue
WRED:	Weighted Random Early Detection
WWW:	World Wide Web

Chapter 1

Management Challenges in Networks and Distributed Systems

1.1 Introduction

Systems management represents the set of activities necessary to ensure that information systems function according to user requirements and objectives [Mof94]. Commonly, large networks have a number of types of hardware and software components with specific features. These features differ from component to component and cover many capabilities related to capacity, scalability, speed of connection etc. In order to be managed these components are modeled by so-called managed objects (MOs). Figure 1.1 below depicts a common way in which management activity is viewed [Osi91].

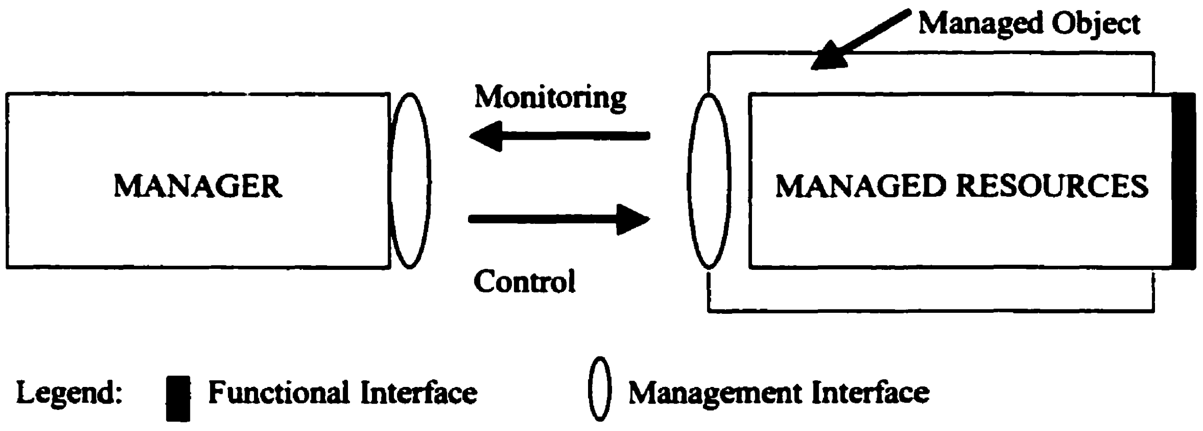


Figure 1.1 Management Activity

Here, the manager interacts with a managed object through a management interface. Managed objects represent abstractions of managed resources. Managed resources have functional responsibilities that are fulfilled through interactions via a functional interface [Put95].

Managers can be looked upon as objects with management responsibilities and may, at the same time, be function of higher-level management. Managers manage MOs by various actions such as monitoring MOs behavior, making management decisions based on the monitored behavior and modifying MOs behavior through management operations [Mof94].

Systems management deals with activities that ensure the meeting of the functional service levels required by the users of the system. This is not concerned directly with the functional activities of the managed systems and, in this sense, it is a meta-activity.

In contrast to Figure 1.1, real life management systems consist of a large number of managed and managing objects. Several problems present in large-scale management systems have been identified [Slo94]:

- Central management of large-scale systems is a difficult task because these systems cross-organizational frontiers.
- Multiple managers are often required to deal with large scale systems; these managers could be hierarchically organized, but even such a ranking can introduce problems with the delegation of authority and responsibility.
- Managed objects can fall under the responsibility of more than one manager and, in such cases, conflicting management requirements from different managers can arise.
- Large-scale systems require the existence of grouping mechanisms for MOs in order to deal with such a high number of them.
- For very large management systems, it is important to automate the management process as much as possible in order to assist human managers in the management of systems.

Mechanisms that simplify the management process are needed in large scale management systems. These mechanisms include [Put95] [Mas93]:

- Increasing the level of abstraction at which interactions occur so that managers can interact with groups of MOs rather than with separate MOs.

- Using management policies instead of controls so that users are enabled to specify the required service levels rather than specifying how these levels can be achieved.
- Automating the process that captures and transforms management policies to control operations.

The changing business requirements of companies, together with the increased scale and complexity of mission critical applications, place extra pressure on system developers. Rapidly delivered information systems that dynamically adapt to change are the most preferred [Slo94][Poo94].

Policy driven management systems cope with policy changes. A management policy is a relationship between a set of managers and a set of managed objects; this relationship includes obligation and/or authorization rules for managers to perform activities on managed objects. The following are managing and managed entities: managers, managed objects, policies, functional systems and subsystems and services provided by support environments and management platforms [Put95]. All these entities are viewed as objects.

As illustrated in Figure 1.2, management policy can be used as a mechanism to capture the goals and requirements from the users. This captured information needs further transformation into management operations that, in turn, serve to influence the behavior of managed systems in order to satisfy user requirements.

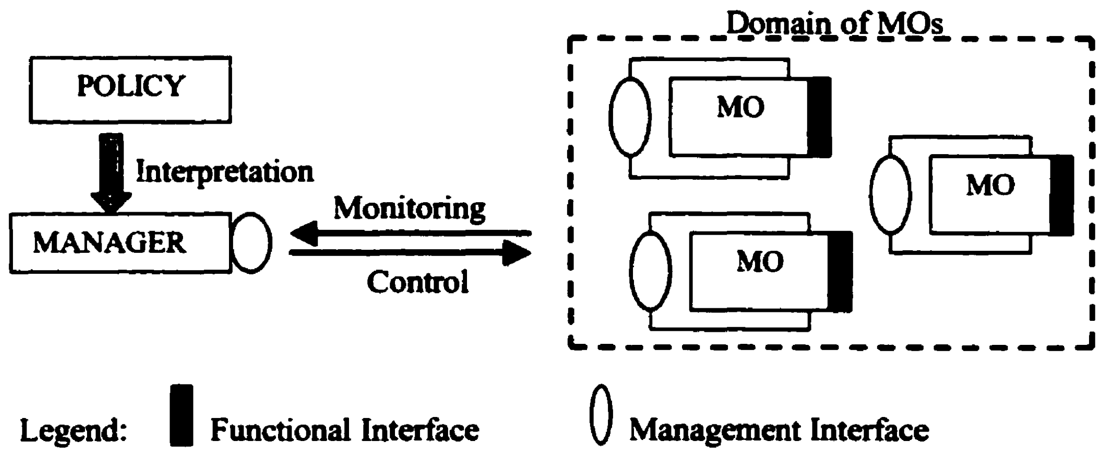


Figure 1.2 Policy in systems management

Large management systems are divided into domains in order to ease their management. Domains are groups of MOs to which a common policy applies and which act as a naming construct for objects (unique name space). They can also be used for the specification of viewpoints (partial behavior) with specific emphasis on systems [Slo94]. Management domains represent groups of objects for the application of a common management policy. On the other hand, functional domains are groups of objects to structure functional systems [Put95].

Policy management systems exist everywhere in companies and organizations. Usually, there is a very close link between policy statements and system code, such that the policies themselves are rarely articulated [Poo91]. This situation gives rise to some problems, as briefly presented below [Ngu92][Mas93][Mof94]:

- It becomes very difficult to capture, store, query and modify policies in a structured manner.
- Managers do not interpret policies in a consistent way, so it is difficult to implement re-usable managers.
- Inconsistencies and conflicts can easily arise.
- It makes it very difficult to modify policies because these changes have to be made directly to the system's code.
- It makes it very difficult to modify policies dynamically and to forecast the effects of policy changes.

A solution to the above mentioned problems is to treat and implement policies as a separate concern [Mof94][Ngu93][Mcb91]. This new policy modeling comes with various advantages such as (i) policies are recognized as deliberate, (ii) they are well-defined, (iii) it is easier to manage them and (iv) it is easier to assess their correctness. At the same time, management policies can easily be modified and interpreted. Reusable managers can also be implemented.

Policy management services should provide mechanisms to ensure the following actions [Slo94][Ngu93]:

- Create, modify and delete policies;
- Represent and interpret policies;
- Store and retrieve policies;
- Negotiate the outcome of conflicting policies, and
- Communicate new policies or modifications done to existing policies to concerned managers.

A policy model should be consistent, should capture policies at a high level of abstraction and should allow high-level policies to be transformed into concrete plans to meet the necessary requirements [Mas93]. Mechanisms to refine abstract, high level policies, should exist, and automation of this process should be considered as extensively as possible [Mcb9][Slo94]. Some actions can easily be automated. These include [Put95]:

- The capturing of requirements from end system users in order to shorten the gap between the policy requirements and the operations needed to enforce these requirements;
- The detection of incomplete information in user requirements statements;
- The detection of managers and MOs to which certain policies apply and the distribution of these policies to concerned managers;
- The detection and resolution of policy conflicts, and
- The transformation of policy statements into management operations.

Relationships between policies should also be represented in order to allow human managers to assess if stated policies have been satisfied [Mof94]. These relationships between policies are very often hierarchical in nature, where the authority is appointed downward. Policy hierarchies are characterized by partitioned targets, refined goals and delegated responsibilities [Mof94][Mas93]. If a less hierarchical nature of organization is adopted, a particular MO can fall under the management of more than one manager and policy conflicts may arise.

Various strategies can be used to deal with policy conflicts. The two main approaches are either avoiding policy conflicts or resolving them when they occur. A combination of avoidance and detection mechanisms can be used for better results. Another possibility is to allow a certain degree of policy conflict, as detailed in [Mof94].

1.2 Managing across Private Networks and Internet

Many proprietary architectures tried to solve the above problems [Terp92]. IMB's OMA (Open Management Architecture) is tailored for managing SNA (Systems Network Architecture) networks, as well as Token Ring LANs. Mainly, the fault functionality is covered. The AT&T's UNMA (Unified Network Management Architecture) is an OSI (Open System Interaction)-based architecture, structured in three levels: Network Elements, Element Management Systems, and Integrated Management Systems. The OSI CMIP (Common Management Information Protocol)-based NMP (Network Management Protocol) really permits the integration of vendor management systems. DEC's EMA (Enterprise Management Architecture) has an object-oriented information model and uses an architecture similar to OSI Management. The concept of domain is used to divide network management solutions. There is a critical problem to integrate such private solutions in order to achieve automatic management. However, many attempts exist. The goals of standardization, industry, and research groups that propose information models, management protocols, and management architectures are derived from the imperious need to facilitate the management of networks and distributed systems. The final purpose is to develop management systems powerful enough to automatically monitor and control the behavior of managed systems. Management systems require support platforms offering particular services to achieve the desired automation. The management tools presented above allow a system operator to act on the real managed and managing systems, as well as on their object-oriented representation.

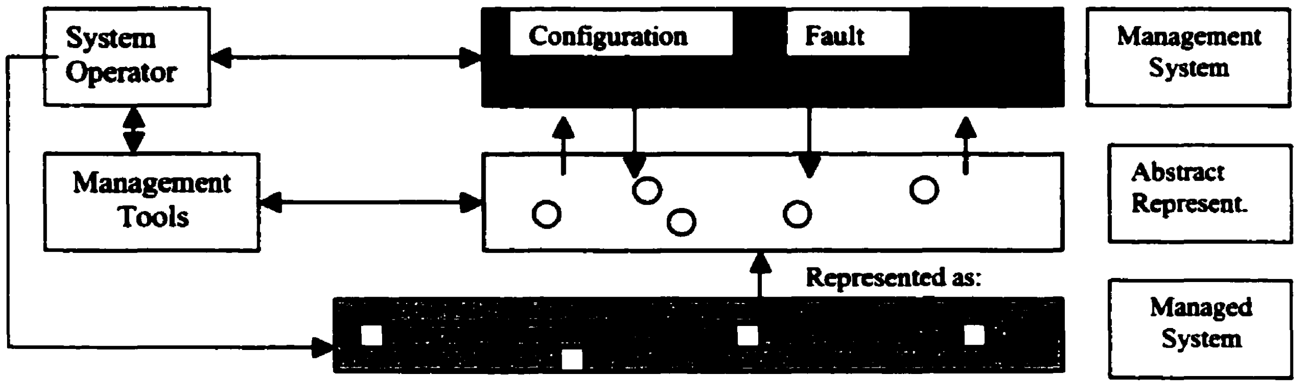


Figure 1.3 Managed and Managing systems

The generic framework of a managed system and its managing system, presented in Figure 1.3, shows different management interactions between system operators, managed, and managing systems. Real components of a distributed system and of its management system are represented as objects. The management system interacts with this representation to perform various management tasks conforming to management functionalities, such as fault recovery, security, and reconfiguration. A system operator can directly interact with the management system or real components, and, across specialized tools, with the abstract representation. Management functionalities may be performed either by the system operator directly on managed systems or across specialized management tools, or automatically, by management systems. However, even in the last case, a minimum cooperation with the system operator is unavoidable. The current tendency is to clearly define management concepts and unify heterogeneous proposals. The final goal is to use these concepts to design management systems, able to automatically apply management decisions. To develop such applications, the distribution of processing, storage, and management must be considered. Consequently, support platforms offering services to manipulate objects belonging to managed and management systems are required. Such platforms must offer services to monitor and control interactions between objects, according to the nature of applications.

1.2.1 Proprietary management tools

Many vendors developed general proprietary management tools to deal with various aspects of the network management area. For example, Hewlett-Packard's generic network management solution, HP OpenView Express 2.0 is an integrated version of several existing HP products, including ManageX for application and systems management, Network Node Manager 250 for NT for network management, and OmniBack II for NT for managing storage. Sun's Management Center tool proposes a large spectrum of proprietary management tools, covering the aspects of network bandwidth management, network fault/performance management, and system management [Sun98]. Tivoli's management solution offers a comprehensive IT (Internet

Technology) management system for any environment regardless of size, complexity, or growth rate [Tiv97].

1.2.2 Managing the Internet

Web technologies started to be used in IP network management as early as 1993 [Fla99]. Today, many network equipment vendors such as Cisco, Nortel Networks and 3Com, embed HTTP (Hypertext Transfer Protocol) servers in their new network equipment. The concept of embedded management application was introduced [Sun95], including the advantages of using HTTP rather than SNMP (Simple Network Management Protocol) to transmit data between managers and agents. The idea is to transform a vendor-specific management GUI (Graphical User Interface) that has to be ported to different management platforms and operating systems into an applet that can run everywhere. In addition to this proposal, one can use Java servlets and Remote Method Invocation (RMI) to open persistent sockets between applets and servlets.

With all these new possibilities in mind, two general network management paradigms have been proposed: the *pull model* [Sun95][Fla99] and the *push model* [Fla99]. In software engineering, the pull and the push models refer to two approaches for exchanging data between two distant entities.

(a) The Pull Model

The *pull model* uses the request/response paradigm. The client, that is, the manager, sends a request to the server, which gives an answer either synchronously or asynchronously. This model can be used for both ad hoc and regular management. In ad hoc management, a network manager connects to a network device via some GUI, retrieves the desired data and closes the connection. In regular management, continuous data collection, network monitoring and event handling take place.

In the context of ad-hoc management, the pull model can be used in various circumstances as described further. One approach is to code the vendor-specific management GUI as an applet [Sun95]. The uploading of the applet by the Web browser proceeds as follows. The HTTP server from the agent retrieves its vendor-specific management applet from local storage and sends it to the Web browser. After the Web browser uploads the applet, either SNMP or HTTP can be used for the communication [Fla99]. Another approach [Fla99] uses generic management GUIs coded as applets and relies entirely on Web technologies. A generic GUI supports generic MIBs (Management Information Base) such as MIB-II, RMON MIB (Remote Monitoring MIB), or ATM MIB (Asynchronous Transfer Mode MIB) [Mar98].

In the context of regular management, automation is extensively considered. Data polling and event handling are also being taken care of. In the case of data polling, for example, there are well-defined steps to be performed in order to integrate two separate management platforms:

- integrate a Web browser in the network management platform;
- replace all the GUIs of the network management platform [Mar98] with applets;
- make the data repository independent of the network management platform;
- implement data polling with HTTP, using a servlet, and
- migrate reports generation to Web technologies, using another servlet.

At this point, data collection and monitoring rely entirely on Web technologies.

(b) The Push Model

In contrast to the pull model, the *push model* uses the publish/subscribe/distribute paradigm. Agents first advertise what MIBs they support and what SNMP (Simple Network Management Protocol) notifications they can generate. A human administrator subscribes the manager to the data she or he is interested in, specifies the desired receipt of data frequency and then disconnects. The agents push the data to the subscribed managers either by following a pre-established schedule or asynchronously via SNMP notifications.

In the case of the push model, the communication is initiated by the agent instead of the manager, as was the case with the pull model. To deal with this issue, three technologies can be used: HTTP, sockets, or RMI [Fla99].

Java servlets can also be effectively used in Web-based messaging management [Jon99]. Java has proven to be an effective support for providing an integrated management function. Java and Web-based management schemes confer flexibility in dealing with custom management requirements, security functions and cost.

Networks today are composed of many interconnected heterogeneous resources. Network management standards are essential in consistently managing these resources and they are an important step towards integrated network management. It is very difficult for the end user to manage network resources in a single and consistent way because many network resources come with proprietary software tools running only on specific platforms and powerful, general-purpose network management tools are complex and costly [Der96].

The World Wide Web (WWW) is a new way to provide wide access to complex software applications. It contains general-purpose information either distributed or contained in a single large database. The Web can be used in the complex situations of network management, such as changing network topology [Der96]. Specific management problems can be solved by combining technology presented with the power of the WWW. Following are the major advantages of using the Web when dealing with network management problems:

- it is an open and established technology;
- it has a limited configuration;
- it is based on inexpensive and standard web technology (Liaison and the GDMO/ASN.1 (Guidelines for Definition of Managed Objects / Abstract Syntax Number 1) Search Engine are easily accessible [Der96]);
- uses a machine with limited power to manage network resources;
- it has reduced cost since users can share a single Liaison or GDMO/ASN.1 Search Engine, and
- additional security is provided by the Web in addition to the CMIP/SNMP security mechanisms.

Web-Based Enterprise Management (WBEM) and OSI management are two management approaches that share many common concepts. Both approaches use a manager/agent paradigm. One agent maintains a Management Information Base (MIB) which is a set of MOs, and a management service and protocol give access to the agent's

MOs [Fes99]. Both models take an object-oriented approach to information modeling. Despite these similarities, certain differences exist, as outlined below.

Topics	OSI versus DMTF(Distributed Management Task Force)	
	OSI	DMTF
Specification Languages	- GDMO (MOC, MO instance) [Iso1]. - GRM (General Relationship Model) [Iso2]. - ASN.1 (data-type) [Iso3].	- MOF (Managed Object Format) (MOC MO instance, data-type) [Dmtf].
Specification Unit	- Module with OSI global naming.	- Schemas.
Inheritance	- Multiple.	- Simple.
Attribute scope	- Outside MO class, grouped in packages optionally added to class.	-Directly defined in MO class context.
Data-type (attributes, parameters, reply)	- All possibilities of ASN.1 (list, sequence).	- Limited set of basic data types (integer, boolean).
Basic containment relation and naming	- Hierarchical Name binding between MO instances using one attribute per Instance Distinguished Names.	- Name-space (hierarchical directory structure) for MOC and MO instances. - Many key attributes per instance.
Relationship between MOs	- Pointer attribute. - GRM class and instance. - Management action and/or operation.	- Special managed objects containing references to other objects (association qualifier).
Specification repository	- Dedicated MIB definition and agent [Itu].	- MOF MOC definition are stored in name space.
Management service/protocol	- CMIS/CMIP [Cci91]. Basic services, for instances: creation, manipulation, and deletion. - Most of the services can apply on a set of instances specified by scoping rules and filters.	- HMMP(HyperMedia Management Schema). Basic service for instances and classes creation, manipulation, and deletion. - SQL-like query language (HMQL).
Event	- GDMO specification Management service available (Event Report Service).	- none.

Table 1.1 OSI versus DMTF.

1.3 Quality of Service (QoS), Type of Service (ToS), Class of Service (CoS)

Nowadays, the majority of traffic does not come from within a workgroup, but rather from outside it and from over the Internet. This situation arises because of several reasons [Int00]:

(1) Companies are moving mission-critical applications from local systems to enterprise-wide server clusters;

(2) Inter-network global communications are increasing to deal with the expansion of corporate businesses and technology partnerships, and

(3) Users are increasingly using web-based technologies for research, product search and evaluation and communication.

At the same time, both volume and type of data have expanded due to:

(1) higher speed PCs generate traffic faster than they used to

(2) Fast Ethernet at the workstation, Gigabit Ethernet in the backbone and Layer 3 switching technology moving packets faster through the network

(3) increasing use of "killer" applications such as Voice-Over IP (VOIP) requiring faster service than the one presently in place.

The convergence of different, more bandwidth demanding, and faster applications stress the networks, which become severe bottlenecks and inhibit higher service quality for the more demanding applications. As a consequence, one obtains delays of mission-critical data, jittery video quality or choppy voice over IP communications. Core routers can be arranged to run fast routing mechanisms.

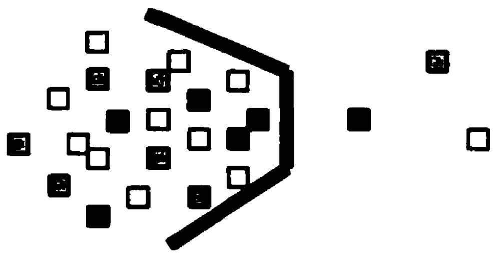


Figure 1.4 Increased traffic of more demanding applications bottleneck edge routers

As a response to service degradation due to bad response to user related scalability, unexpected failures, or user profile changes, managers on campus used additional bandwidth and faster equipment, but this was a costly solution. Outside the campus, such decisions were not under manager's control and the cost was much higher. At this point, the idea of prioritizing network resources to traffic requirements emerged, and this was under the manager's control. To implement this kind of management, policy-based management mechanisms emerged as a new paradigm. Quality of Service (QoS), Type of Service (ToS), Class of Service (CoS) mechanisms, such as Differentiated Services (DiffServ), as well as Integrated Services (IntServ) are at the base of policy-based network management and prioritizing resources to requirements.

(a) Quality of Service (QoS)

Quality of Service (QoS) mechanisms provide the necessary level of service to an application in order to maintain an expected quality. For mission-critical applications, QoS means guaranteed bandwidth and zero frame loss. For a Voice over IP application, QoS means guaranteed frame latency. The detailed control provided by QoS overwhelms the network infrastructure, especially the boundary elements (e.g. edge routers). Each network device must keep an entry for each flow in its forwarding table.

Some examples of protocols that deliver level of service by application flow are ATM, Frame Relay, MPLS (Multi-Protocol Label switching) and RSVP (Resource Reservation Protocol). RSVP delivers end-to-end service by reserving bandwidth and resources along a particular path. It is implemented in some routers and in network layer switches. Nevertheless, complete end-to-end quality of service implementations are difficult and costly to be achieved [Int00].

(b) Class of Service (CoS)

Class of Service (CoS) mechanisms map multiple flows into a few service levels to reduce complexity. Resource allocation is then done according to these service levels and flows can be cumulated and forwarded according to the service class of the packet. CoS applies bandwidth and delay to different classes of network, a matter much easier achieved than QoS control. As the traffic grows, it continues to be managed based on a few service levels.

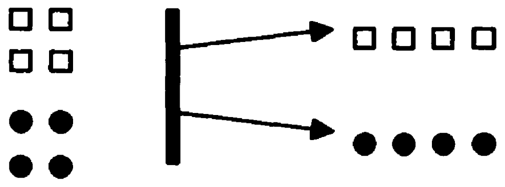


Figure 1.5 Class of Service maps multiple flows to a few service levels

(c) Type of Service (ToS)

One common CoS mechanism is Type of Service (ToS) prioritization, at layer 3 of the OSI layered network model [Int00]. CoS can be in Ipv4 or Ipv6.

Type of Service adds to the Ipv4 header 3 precedence bits which describe 7 different priority levels. There are 5 unimplemented bits. Boundary routers and ToS-enabled layer 3 switches map these precedence bits to forwarding and dropping behaviors. In Figure 1.6, the ToS octet in the Ipv4 header is further detailed. Bits 0 to 2 represent the "precedence", bit 3 represents the "delay", bit 4 represents the "throughput", bit 5 is the "reliability", and bits 6 and 7 are left for further use.

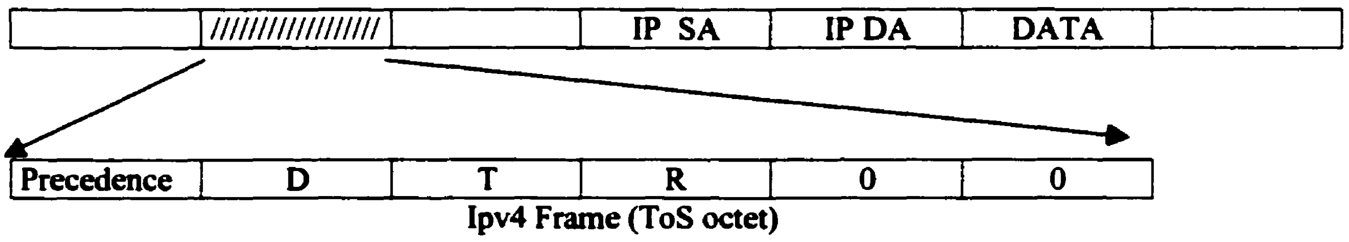


Figure 1.6 ToS provides 3 prioritization bits in the Layer 3 header

1.3.1 QoS parameters

When traffic passes through a network connection, the receiver can, theoretically, obtain the traffic identically as the sender sends it. However, since the network is not perfect, the communication is affected by different factors. In general, these factors are referred to as QoS parameters, which are measured, evaluated and monitored over a certain time period.

One method to classify QoS parameters is depicted in Figure 1.7. This taxonomy clearly identifies different classes of QoS parameters [Sab97]. Policies and metrics form the two main categories. Policies are further divided into management and levels of service, and are responsible for network management. Metrics are grouped into security levels, performance and relative importance. Security levels specify the data security level that needs to be provided to the application. Performance refers to parameters related to the performance of the network services. Relative importance specifies the price that a user is willing to pay for a QoS.

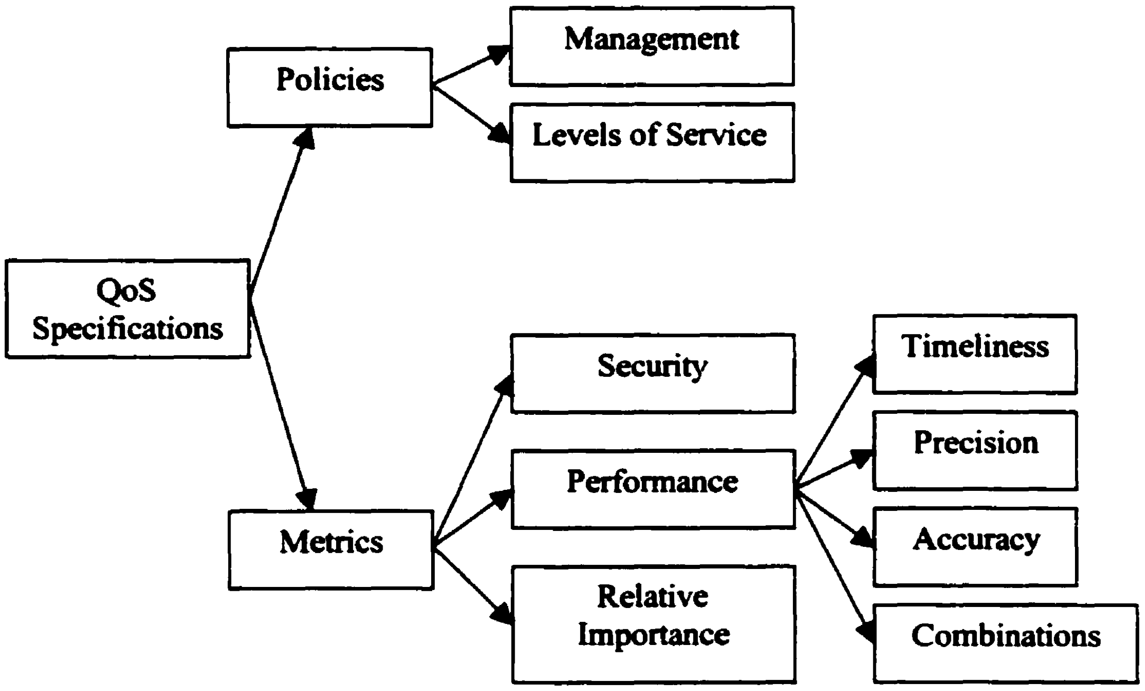


Figure 1.7 Taxonomy for QoS specifications

Network communication expects the QoS parameters to be specified in order to determine whether or how much QoS can be provided. One single parameter is not enough, but a set of such QoS parameters is required. Therefore, there is a need to have a formal way to evaluate QoS parameters. Following, we present some QoS parameter definitions for general networks together with their formulas. They are grouped in (a) basic parameters, including those which are the most important for network performance and (b) other parameters.

(a) Basic Parameters

The four most important QoS parameters reflecting network performance are: bandwidth, latency, delay jitter and loss rate. The calculation of these parameters is unavoidable when evaluating the system's performance with respect to QoS.

QoS of a connection is evaluated by many parameters, which are measured at each component along the connection path. If the network entities cooperate in a linear configuration, that is, from server QoS to network QoS to terminal QoS, then the end-to-end QoS can be calculated based on the QoS characteristics of each component according to the QoS parameter type. In the following formulas, the end-to-end QoS parameter values ($P_{\text{end-to-end}}$) is calculated from the QoS parameters of the i-th component, where $i = 1,2..,n$.

(i) Bandwidth

Bandwidth, or throughput, represents the maximum number of bytes that can be transmitted over a connection in a certain time interval. A bandwidth of 15 million bits per second (Mbps) means that the network is able to deliver 15 million bits each second. Another way of viewing bandwidth is in terms of how long it takes to transmit one bit of data. In the example used, it would take 0.15 microseconds (μsec) to transmit one bit of data [Wol98].

End-to-end bandwidth or throughput is the minimal throughput of links and nodes along the path. The minimum throughput at a link is the limiting factor that determines the capability of a connection.

$$\text{Throughput}_{\text{end-to-end}} = \textit{Minimum}\ (\text{Throughput}_i), \text{ where } i = 1,2\ldots, n$$

The minimal bandwidth a network offers can not be less than the bandwidth needed to satisfy real-time traffic.

$$\textit{Minimum}\ \text{Bandwidth}_{\text{path } n} \geq \text{Bandwidth}_{\text{request}}$$

For service that does not require real time, the network manager makes use of bandwidth readjustment technologies to balance traffic load and affordable bandwidth [Por97].

(ii) Latency

Latency, or delay, refers to the interval of time between the generation or access of media data and its presentation. Delay is also viewed as the period of time from the moment data is received at an input port and the moment it is sent out on the output port. Time is the only important metric for latency.

Latency is composed of: *propagation, transmission* and *queuing*. Propagation latency is the speed of light transmitting in the media between two ends of a network. It is known that light travels through vacuum at $3.0*10^8$ meters per second and through fiber at $2.0*10^8$ meters per second. Transmission latency is the time a network takes to transmit packets of a certain size under its bandwidth. Queue latency represents the amount of time packets spend in routers before being forwarded onto the output link. Routers generally use the FIFO (First-In-First-Out) algorithm to decide which packet is next to be forwarded [Wol98].

End-to-end latency is the sum of all the delays in nodes on the links along the path from source to destination. The formula used for its calculation is [Haf98]:

$$\text{Latency}_{\text{end-to-end}} = \sum_{i=1}^{n} \text{Latency}_i$$

(iii) Jitter

Jitter is the variance in the delay when processing and transmitting data. It can be smoothened by buffering at the receiver side, but this action increases the end-to-end delay [Wol98].

The end-to-end delay jitter is calculated by adding up all the delay jitters of the nodes along the path. This value can not exceed the service requested jitter [Haf98].

$$\text{Jitter}_{\text{end-to-end}} \leq \text{Jitter}_{\text{requested}}$$

If we consider the jitter to be the difference between the maximum and the minimum delay, the following formula is used:

$$\text{Jitter}_{\text{end-to-end}} = \sum_{i=1}^{n} \text{Jitter}_i$$

If we assume the jitter to be the average deviation of the delay from the average delay, with the delay considered to follow a normal distribution, than the following formula more is appropriate:

$$\text{Jitter}_{\text{end-to-end}} = \sqrt{\sum_{i=1}^{n} (\text{Jitter}_i)^2}$$

(iv) Loss Rate

Loss rate represents the ratio between the number of bits lost during transmission and the total number of bits sent by the source. Data loss occurs when a data segment is transferred with an error, with a duplicate copy or is lost in the network. Loss rate is generally specified by bit-error or packet-error rate, but, sometimes, service providers use additional criteria to determine the capability of delivery on a guaranteed service. For example, if only 97% of the data sent is delivered to the destination, one might consider this to be a high loss rate. This situation most probably arises due to lack of resource, such as bandwidth, so increasing the buffer size is helpful here to increase the performance in terms of packets lost. On the other hand, a larger buffer can store more messages waiting to be sent, which extends the delay in data transmission [Wol98].

Loss rate in a path is function of the loss rate for every connection. The following formula is used for its calculation [Haf98]:

$$\mathrm{Log}\,(1 - \mathrm{LossRate}_{\text{end-to-end}}) = \sum_{i=1}^{n} \mathrm{Log}\,(1 - \mathrm{LossRate}_i)$$

or $$\mathrm{LossRate}_{\text{end-to-end}} = 1 - (1 - \mathrm{LossRate}_1)^* (1 - \mathrm{LossRate}_2)^* \ldots ^* (1 - \mathrm{LossRate}_n)$$

(b) Other Parameters

In addition to the basic QoS parameters presented above, other parameters related to end-to-end communications exist. In this sub-section, we briefly present a few of them.

(i) Cost

The end-to-end cost represents the maximum price a user is willing to pay for a given quality of service in a system where users compete for resources. It is computed by adding the costs on all the network components along the path.

$$\mathrm{Cost}_{\text{end-to-end}} = \sum_{i=1}^{n} \mathrm{Cost}_i$$

(ii) Hop Number

In general, there is more than one path between two points in a given network, with varying number of hops. The route having the minimum number of hops is regarded as the hop-number of a particular end-to-end connection.

$$\text{Number\_Of\_Hops}_{\text{end-to-end}} = \text{Minimum}\,(\text{Number\_Of\_Hops}_i)\,,\ \text{where } i = 1, 2, \ldots, n$$

(iii) Blocking probability of new connections

Blocking probability of new connections represents the ratio between the number of rejected connections and the total number of connections. Because new connections always have lower priority than existing ones, they are always blocked or rejected when the bandwidth becomes scarce. One solution to reduce the blocking probability is to increase the bandwidth, but this implies a higher cost. Algorithms that employ less bandwidth without lowering the performance are preferred as suitable solutions for decreasing the blocking probability of connections.

(iv) Minimum allowed bit rate

The minimum allowed bit rate is the minimum amount of bandwidth required for a connection to operate within acceptable QoS boundaries. This parameter is provided by the user for real-time applications. In the case of applications that don't require real-time, the minimum allotted bit rate is set by the network and represents the available bandwidth.

(v) Bandwidth readjustment probability and Robustness

When not enough bandwidth is available for transmission traffic, bandwidth readjustment is performed based on the available bandwidth in the network. A network connection is robust if it has low bandwidth readjustment probability [Oli95].

(vi) Connection duration

The duration of a connection QoS parameter is the time interval between the issuing of a connection request and the receipt of the confirmation information for that connection. It is used in evaluating the performance of algorithms.

(vii) Connection failure probability

Connection failure probability is the probability that a requested connection is not established within the maximum acceptable time for setting up a connection.

(viii) Release delay

Release delay represents the time interval between the issue of a disconnect request and a correspondence for successful release by the service provider [Hut94].

1.3.2 Special management functions due to Internet

As we have seen, a management system is an application which consists of specialized managing objects playing different management roles, such as performance monitoring, fault detection, or system reconfiguration. These roles are fulfilled based on the information collected by managing objects using management operations on managed objects, and the interpretation of the results of these operations.

In distributed systems, many functionality areas must be considered, as shown in Figure 1.8 below.

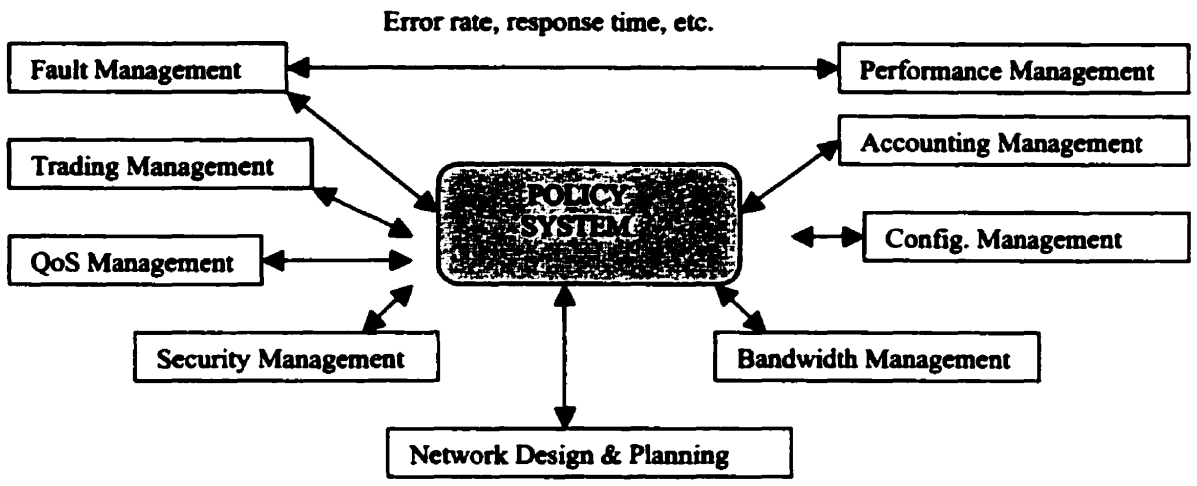


Figure 1.8 Information Exchange between Management Functional Areas

The key functional areas of network management as defined by International Organization for Standardization are: Fault Management, Accounting Management, Configuration and Name Management, Performance Management and Security Management [Sta93]. This functional classification is widely accepted by vendors of standardized and proprietary network-management systems.

(i) Fault Management

In complex networks, the proper functioning of each network component as well as of the system as a whole is mandatory. A fault, as distinguished from an error, is an abnormal condition that requires management actions in order to be repaired. In the case of a fault occurrence, the following actions must take place as soon as possible:

- localize the fault;
- isolate the fault from the rest of the network so that it is prevented from interfering with the proper functioning of the remaining network;
- reconfigure the network so that it becomes able to function appropriately without the faulty component, and
- repair or replace the faulty component to obtain the network's initial state.

Examples of faults are : incorrect operation of a network component or excessive errors. Some types of errors, such as single bit errors, are not normally considered to be faults. Also, error-handling mechanisms of different protocols can compensate for some errors.

From a user perspective, fault resolution should be fast and reliable. At the same time, users expect instant notification of a fault occurrence. To ensure these user's expectations, very rapid fault resolution techniques and reliable fault-detection and diagnostic-management mechanisms are suitable. The presence of redundant network components and alternate routes confer some degree of fault tolerance. After resolving the fault, the fault management is responsible for ensuring that no other problems have been caused by the solved one. The entire procedure should be performed with minimal network performance damage.

(ii) Accounting Management

Networks possess internal accounting procedures for different corporate networks, divisions or cost centers that make use of network services. In the case when no such internal accounting procedures are employed, the network manager must be able to identity the user or the user class that employs network resources because of different reasons briefly outlined below:

- a user or group of users may abuse their access privileges at the expense of other users of the network
- the network manager may assist users that make inefficient use of network resources in improving network performance
- knowledge of the user activity in a network allows the network manager to better plan the network growth.

From a user point of view, the network manager should be able to specify the various accounting information necessary at every node, the time period between sending the recorded information to higher-level management and the algorithms used in calculating changes. Users themselves should have their authorization verified when they access and manipulate information.

(iii) Configuration and Name Management

Network components can be configured to perform many different applications. For example, the same entity can be configured to act as an end-system node, or as a router, or as both. Configuration management takes care of initializing a network and gracefully shutting it down. In the course of network operation, configuration management maintains, adds, and updates the relationships among components and the status of the components.

From the user point of view, the network manager should be able to identify the components of a network and determine their most appropriate connectivity. Other tasks that a network manager should be able to perform, from a configuration point of view, are:

- change the connectivity of the network or network components, and
- reconfigure a network as a result of network performance evaluation, upgrade, fault recovery or security issues.

Some configuration management operations are automated, such as configuration reports periodically generated to users, or as response to a request.

(iv) Performance Management

Components in complex networks should be able to interact with each other and share resources. Many applications require that the communication between network components is within pre-defined performance limits.

Performance management includes monitoring and controlling. Monitoring keeps track of network activity, whereas control makes adjustments to improve network performance. Some performance management issues include:

- the level of capacity utilization;
- the existence of excessive traffic;

- the level of the throughput;
- the existence of bottlenecks, and
- the existence of an increase in response time.

From a user perspective, performance should be precisely known in order to respond to specific user queries. Users expect a good response time from the network. Network managers use performance statistics to help them predict bottlenecks before these actually occur and take the corresponding correcting actions.

(v) Security Management

The tasks of security management include:

- generation, distribution and storage of encryption keys;
- maintenance and distribution of passwords and other authorization controls;
- monitoring and controlling access to computer networks and to network management information from network nodes, and
- collection, storage, examination, enabling and disabling of audit records and security logs.

From a user perspective, security management ensures the protection of network resources and user information.

(vi) QoS, Trading, Bandwidth and Configuration Management

Complex interaction exists between managing objects that are responsible for fault, performance, security, network planning, accounting, trading, naming, QoS, and the objects performing configuration management.

Configuration management interacts bi-directionally with fault management. Fault management uses the system configuration to apply particular fault models to discover a faulty component, to diagnose the fault, and to solve it. If a component must be isolated for a test or maintenance, a reconfiguration is expressly requested. Configuration is the basis to apply diagnostic strategies to correlate system alarms and identify the cause of the alarm [Hou95]. In fact, complex dependencies, expressed as relationships between managed system components, lead to different diagnostic policies. Another case where automation can be applied concerns tests within distributed systems. When a component must be tested by a well-defined test component, represented in turn as a managed object, fault management interacts with configuration management to create the adequate relationship instances. It is clear that models where relationships are not represented, or are poorly represented, do not allow such an activity.

A tied relation exists between security management and configuration management. In particular, authenticated associations are used for providing mutual peer authentication. If the access control policy invalidates a relationship already established, this leads to a reconfiguration of the system. Conversely, when configuration management has to create a relationship instance, and the tentative is aborted because of access rights violation, this information is sent to security management. Security management uses the configuration to ensure security agreements. A reconfiguration can be inhibited, e.g. a connection, if a security agreement is violated.

Configuration management cooperates with accounting management to ensure cost-based agreements, according to the performance of the system components and the requested QoS. Accounting management needs current cost information and may request configuration changes. Configuration management may initiate a reconfiguration when the system performance decreases, according to performance criteria, by using the trading functionality. The trading functionality may initiate a request for a new configuration in different scenarios, e.g. a often-requested service is not available with the desired QoS performance, the performance of some suppliers decreases, or the QoS requested by a customer is not longer satisfied by its current suppliers. As a central management activity, the configuration and reconfiguration management copes with the diversity of software versions, component localities, migration transparency, in order to preserve the achieved QoS, or to guarantee a graceful degradation of QoS.

In the Internet world, nobody's land, other functionality areas induced by QoS, ToS and user profiles must be considered such as congestion management, bandwidth management, special NAT (Network Address Translation) [Nat99] mechanisms. These issues will be addressed in the following section.

1.3.3 Existing Remedies

As already outlined, several issues have to be taken care of for the proper functioning of today's Internet. This section presents possible remedies to network problems related to congestion, bandwidth, security and constraints-based routing.

(i) Congestion

It is known that network congestion is a resource sharing problem. Consider the case of a network lacking effective congestion control. When the network load is small, network throughput generally keeps up with the increase in load until a point in reached where the increase in the throughput is much slower than the increase in the load. If the load keeps increasing to reach the network capacity, the queues on network elements will become full, possibly causing packet drops, and throughput will arrive at its maximum to decrease sharply to a very low value. At this point, the network is said to be congested.

A new taxonomy for congestion control algorithms in packet-switching computer networks based on the control theory has been proposed [Cui95]. The network is viewed as a large, distributed control system in which a congestion control strategy is a control policy executed at each node of the network with the maintenance of stable network conditions as its goal. The authors define a set of criteria for control systems as a taxonomy of congestion control algorithms for packet-switching networks as well as how this taxonomy is applied to the individual characteristics of existing congestion control algorithms. The key element of the proposed taxonomy is the decision-making process of individual congestion control mechanisms.

The main taxonomy categories are:

- *Open Loop Congestion Control Algorithms* – these algorithms control decisions do not depend on any feedback information from the congested locations in the network.
- *Closed Loop Congestion Control Algorithms* – these algorithms take their control decisions based on some sort of feedback from the faulty areas.

All open loop schemes have a continuous activation feature and are based on admission mechanisms that tend to stabilize the arrival of traffic at the source. They can act much faster than closed loop schemes. However, only closed loop schemes are able to distribute indications on resource utilization and traffic conditions in the network.

(ii) Bandwidth

Computer networks are becoming busier and more complex with the increasing amount of information and the various information formats. This situation requires increased bandwidth, a resource which is finite. In today's Internet, the quality and quantity of bandwidth cannot be predicted. Certain applications, such as videoconferencing, require a specific quality of service in terms of both response times and delay. Often the network cannot be used to its full potential, even though it may be sufficient for the requirements placed on it. Bandwidth reservation removes this unpredictability by allowing applications to reserve the bandwidth quality of service they need. Therefore, bandwidth allocation becomes an important component of bandwidth management and control. Many vendors propose solutions such as bandwidth brokers to deal with bandwidth management across networks [Ban00].

(iii) Security

Network security has always been an issue in the field of computing and, in the later years, in the area of personal computing. Accessing the network represents an important security risk. Various solutions have been proposed to deal with different aspects of network security. Here are only a few examples: Radius, IPSec, NAT, firewalls, L2TP (Level Two Tunneling Protocol), etc.

Radius is a distributed security system using an authentication server that can be modified to different kinds of networks. Radius uses an open protocol which makes it easy and efficient to interface with networks having large modem pools. The presence of a large modem pool (Internet Provider Services, universities, etc.) always increases network security problems, because there are more opportunities for people to break in or abuse access privileges. Radius authenticates users through a series of communications between the client and the server. Once a user is authenticated, the client provides the user with access to the appropriate network services [Ful00].

IPSec is a simple version of the emerging Internet IP security protocol. It represents a set of protocols being developed by the IETF (Internet Engineering Task Force) to support secure exchange of packets at the IP layer. The two encryption modes supported by IPsec are transport and tunnel. Transport mode encrypts only the data portion of each packet, but not the header. The more secure Tunnel mode encrypts both the header and the data portions of a packet. At the receiving end, an IPSec-compliant network element decrypts each packet [Isp00].

Network Address Translation (NAT) is a router function that allows address reuse. The idea is that the address inside a stub domain can be reused by many stub domains. NAT is seen as a privacy providing mechanism because the machines on the backbone cannot monitor which hosts are sending and receiving traffic [Ege96].

Internet firewalls are security mechanisms allowing limited access to a site from the Internet, or allowing approved traffic in and out according to a pre-defined plan. They allow one to select the appropriate services to meet business needs, while barring others which may have significant security holes. To succeed in repelling unwanted intruders while still providing access to the outside world, firewalls must meet precise requirements [Fir99].

Another security solution is offered by L2TP, which extends the PPP (Point-to-Point Protocol) model [Sim94] by allowing the layer 2 and PPP endpoints to reside on different devices interconnected by a packet-switched network. A user has a layer 2 connection to an access concentrator and the concentrator then tunnels individual PPP frames to a network access server. L2TP uses two types of messages: control messages and data messages. Control messages are used in the establishment, maintenance and clearing of tunnels and calls. Data messages are used to encapsulate PPP frames being carried over the tunnel [Tow99].

(iv) Constraints-based routing

The first step in constraint-based routing is to find a path from source to destination. One condition that has to be satisfied is that the available bandwidth of all network elements in the path has to be larger or equal to the requested bandwidth. To find the shortest path from a source to a destination, one can use different shortest-path algorithms such as Dijkstra's shortest path algorithm or Bellman-Ford shortest path algorithm [Cor89]. One method to select an acceptable path is to eliminate all paths having the residual bandwidth smaller than the requested bandwidth and to choose the shortest of the remaining paths.

A method of choosing an acceptable path to support dynamic routing with QoS constrains is available [Nou97]. The proposed algorithm builds a graph reflecting the change in the network, and then constructs a subnet graph containing only acceptable paths. Next, the algorithm builds the QoS constraint routing table using the subnet graph. Based on the QoS parameters, the algorithm selects the best three paths having the required QoS parameters in the routing tables from source to destination. Multi-parameter applications use a k-shortest path algorithm [Por97] (for example, Double-Sweep Algorithm) to find the k best paths. A recursive procedure is used to select the next offer in a routing table when the current connection request is refused. This dynamic algorithm takes into account multiple QoS parameters and is able to select the best path satisfying these QoS requirements from source to destination independent of specific network technologies.

1.4 Integrated Services vs Differentiated Services Frameworks

Today's Internet Protocol (IP) default behavior in the absence of QoS is known as best-effort service. The TCP (Transmission Control Protocol) / IP nodes make their best effort to deliver a transmission, but they will randomly drop packets in the event of managing the bandwidth or assigning priority to delay-sensitive packets [St299]. The "intelligence" resides in the end-entities while the network remains relatively simple [Sal84]. With the extensive growth of the Internet in the last decade, best effort service proved to be well scalable. Network capacity may be exceeded with the increase in service demands, but service is not denied, instead, it degrades gracefully. This results in a variance in delivery delay (jitter) which does not affect Internet applications such as e-mail, file transfer or Web applications. However, real time applications such as IP telephony require a jitter-free transmission [Stl99].

Increasing the bandwidth is not sufficient to avoid jitter, although it is a necessary first step. The second step would represent the insertion of some intelligence within the network entities so that they become able to distinguish traffic that can tolerate jitter, delay and packet loss from that with strict timing requirements. The goal of protocols supplying QoS is to manage bandwidth so that it is used to provide some control and predictability for different kinds of application requirements.

Taking a satisfaction approach, there are two basic types of QoS: (i) Resource reservation and (ii) Prioritization. In Resource Reservation (integrated services), network resources are distributed according to an application's QoS request and this mechanism is under bandwidth management policy. In Prioritization (differentiated services) network traffic is classified and network resources are given to each class of traffic according to bandwidth management policy criteria. Network elements give preferential treatment to more demanding applications.

Applications, network topology and policy dictate which type of quality of service is most appropriate. Many QoS protocols have been designed to fit together in various architectures and to deliver end-to-end quality of service. Following are the most common:

- ReSerVation Protocol (RSVP);
- Differentiated Services;
- Multi Protocol Labeling Switching (MPLS), and
- Subnet Bandwidth Management (SBM).

1.4.1 End-to-end Integrated Services (IntServ/RSVP)

The IETF Integrated Services Working Group is working to develop standards that dictate how application services define their QoS requirements, how routers learn this information, how to test and validate that the contracted QoS is maintained. In Integrated Services (IntServ), each network element has to identify the coordinated set of QoS control capabilities it provides the information it requires and the information it exports. Routers that support Integrated Services have to classify packets according to some fields specified in the policies. In addition, they have to maintain state information for each individual flow [St299].

IntServ are of two different types:

- *Guaranteed*: they ensure the required bandwidth availability and provide strict bounds on the end-to-end queuing delays, and
- *Controlled Load:* they provide a service equivalent to that provided by best-effort networks under unloaded circumstances, but can not guarantee the strict bounds that Guaranteed services provide.

The ReSerVation Protocol (RSVP) is a signaling protocol that enables IntServ by ensuring the reservation and control of network resources. This is the most complex of all the QoS technologies for both hosts and network elements. RSVP provides the highest level of QoS in terms of service guarantees, granularity of resource allocation and feedback to QoS-supporting applications [Stl99].

Figure 1.9 presents an overview of RSVP.

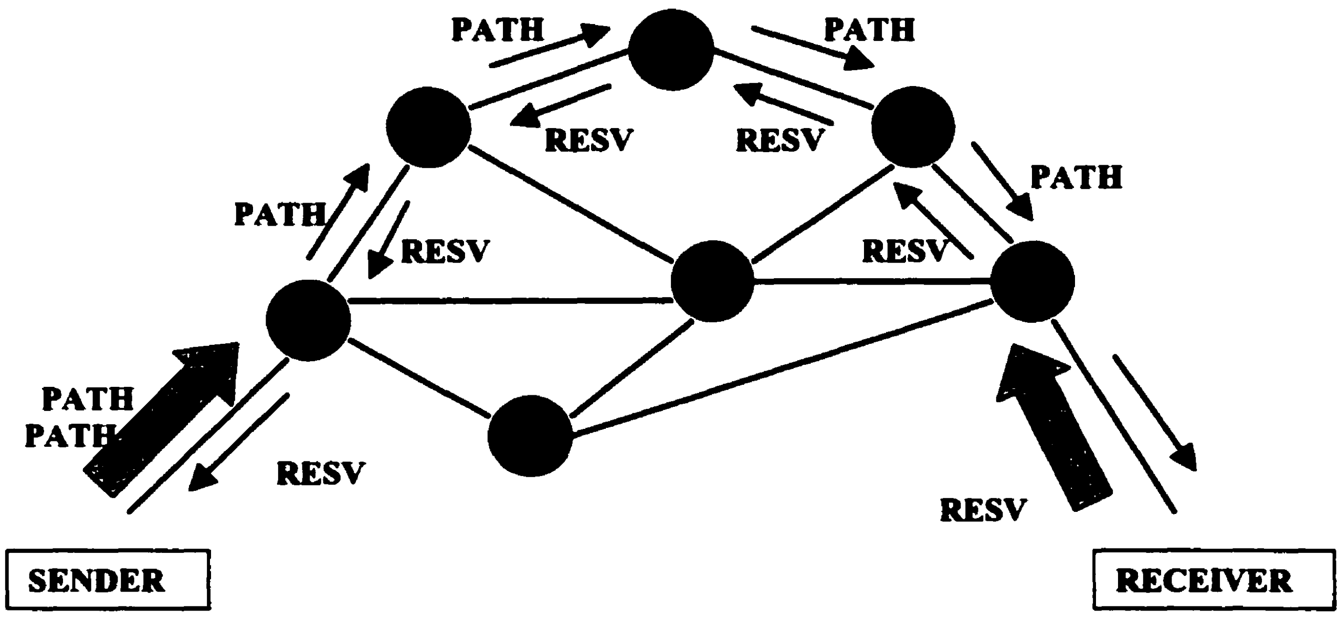


Figure 1.9 RSVP is used to establish a resource reservation between sender and receiver.

In RSVP, the sender sends a PATH message that contains the *traffic specification (TSpec)* in terms of bandwidth, delay and jitter to the destination address. Each router enabling RSVP along this downstream route establishes a path-state with the previous source address of the PATH message. This will represent the next hop upstream towards the sender. The receiver sends a RESV (Reservation Request) message upstream that contains the TSpec, a *request specification (RSpec)* indicating the type of IntServ requested, and a *filter specification (filter spec)* identifying the transport protocol and port number. Together, the RSpec and the filter spec represent the *flow-descriptor* used by routers to identify each reservation. In the upstream path, each RSVP-router uses the admission-control process to authenticate the request and allocate the required resources. An error is returned back to the receiver if the request can not be satisfied. In the case of a request that can be satisfied, the RESV message is sent upstream to the next router. If the router that is the closest to the sender accepts the RESV message, it sends a confirmation message to the receiver. When the sender or the receiver ends an RSVP session, there a tear-down process is initialized destroy the connection created [Stl99].

Although Integrated Services provide the highest level of granularity and the best guarantees for QoS, there are some drawbacks. In particular, the complexity and overhead of RSVP is damaging for many applications or portions of the network. In these cases, methods that provide less granularity and guarantees in terms of QoS are needed. One such a method is Differentiated Services (DiffServ), presented further.

1.4.2 Differentiated Services

Differentiated Services (DiffServ) define a simple method of classifying services of a number of applications [Stl99]. Differentiated Services are not based on priority, application or flow, but on the possible observable forwarding behaviors of packets, named *Per Hop Behaviors (PHB)* [Int00]. The PHB provides a particular service level (bandwidth, queuing, and dropping decisions) according to the network policy (Figure

1.10). Currently, there are two standard PHBs defined that represent two service levels or traffic classes:

- *Expedited Forwarding* (EF): It minimizes delay and jitter providing the highest level of aggregate quality of service. It has a single DiffServ value. All traffic that exceeds the profile defined by local policies is discarded [Jac99]
- *Assured Forwarding* (AF): It has four classes and three drop-precedences within each class, for a total of twelve DiffServ values. Traffic that exceeds the profile defined by local policies is not necessarily dropped, but has a decreased probability of delivery [Hei99].

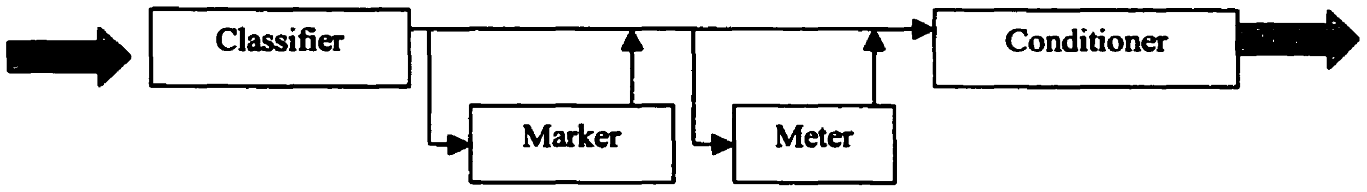


Figure 1.10 Differentiated Services Architecture

PHBs are applied by the *Conditioner* to traffic at a network border entry according to pre-determined policy criteria. The *Marker* can optionally mark the traffic at this point so that it can be routed according to the marking and unmark it at a network border exit point. The *Meter* simply records statistics and it is also optional.

DiffServ makes the assumption that Service Level Agreements (SLA) exist between neighboring networks. A SLA defines the policy criteria and the traffic profile. Any traffic out of profile has no guarantees as it crosses the network. The policy criteria used may include source and destination addresses, transport protocol, port number or time of the day [Stl99].

When applied, the protocol mechanism that the service uses is the *Differentiated Services Code Point (DSCP)* bit patterns, in the IP header. This octet maps to a particular PHB, hence classifying the packet service level. The DSCP replaces the ToS octet in the Ipv4 header and the Class Octet in the Ipv6 header, as depicted in Figure 1.11 below. There are a possibility of 64 different classifications for service levels (8 bits) but only the first 6 bits are currently in use. Two bits are reserved for future expansions. The DSCP retains backward compatibility with the three precedence bits so that non-DS compliant, ToS-enabled devices will not conflict with the DSCP mapping.

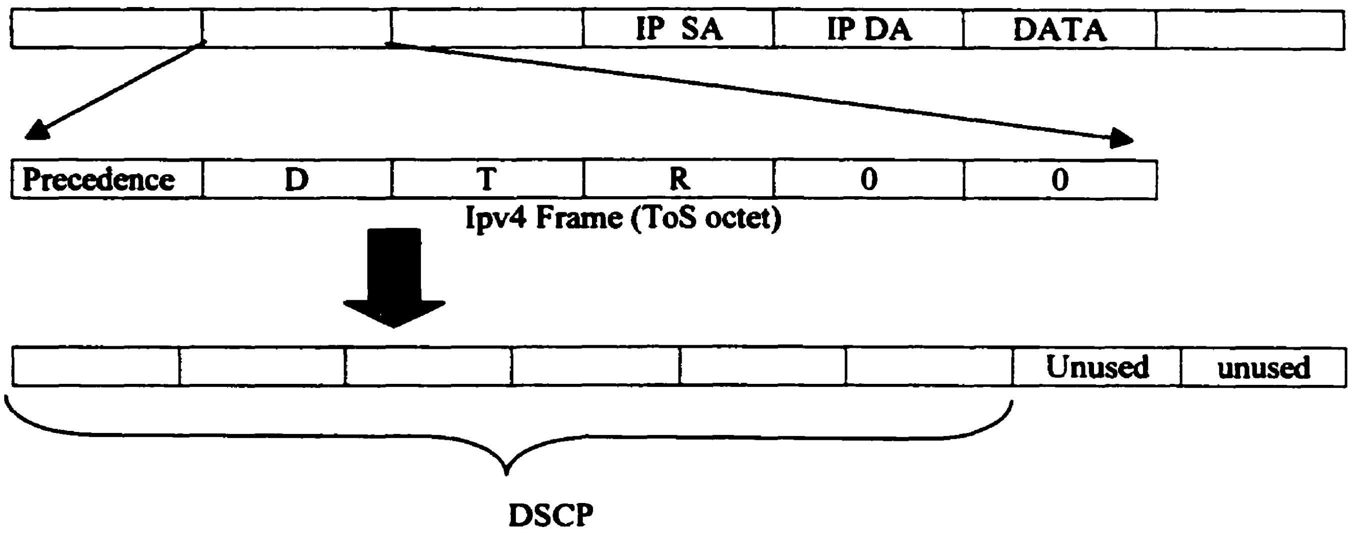


Figure 1.11 The DSCP replaces the ToS octet in the Ipv4 header and the Class octet in the Ipv6 header

Differentiated Services are based on rules, so they represent a good strategy to be used in policy-based network management. The idea is to keep the current network technology and resources and manage networks by appropriate network policies.

As mentioned previously, different kinds of traffic can be marked by the *Marker* for different kinds of forwarding based on network policies. Resources can then be allocated according to the marking and the policies. As an example, mission-critical messages can be encoded with a DSCP bit pattern that indicates high bandwidth and zero frame loss. On the other hand, Web browsing could be encoded with a DSCP bit pattern indicating routine traffic handling.

The power of DiffServ resides in its simplicity to prioritize traffic. If DiffServ uses RSVP parameters to identify and classify constant-bit-rate (CBR) traffic, it is possible to establish aggregate flows that could be directed into fixed bandwidth pipes. In this case, resources can be shared and guaranteed services are still provided [Stl99]. This particular situation is further described while presenting the combination between IntServ and DiffServ.

1.4.3 Combining IntServ and DiffServ

Figure 1.12 below represents the model under development within the IETF and shows how QoS technologies combine together to provide End-to-End and Top-to-Bottom QoS [Ber99].

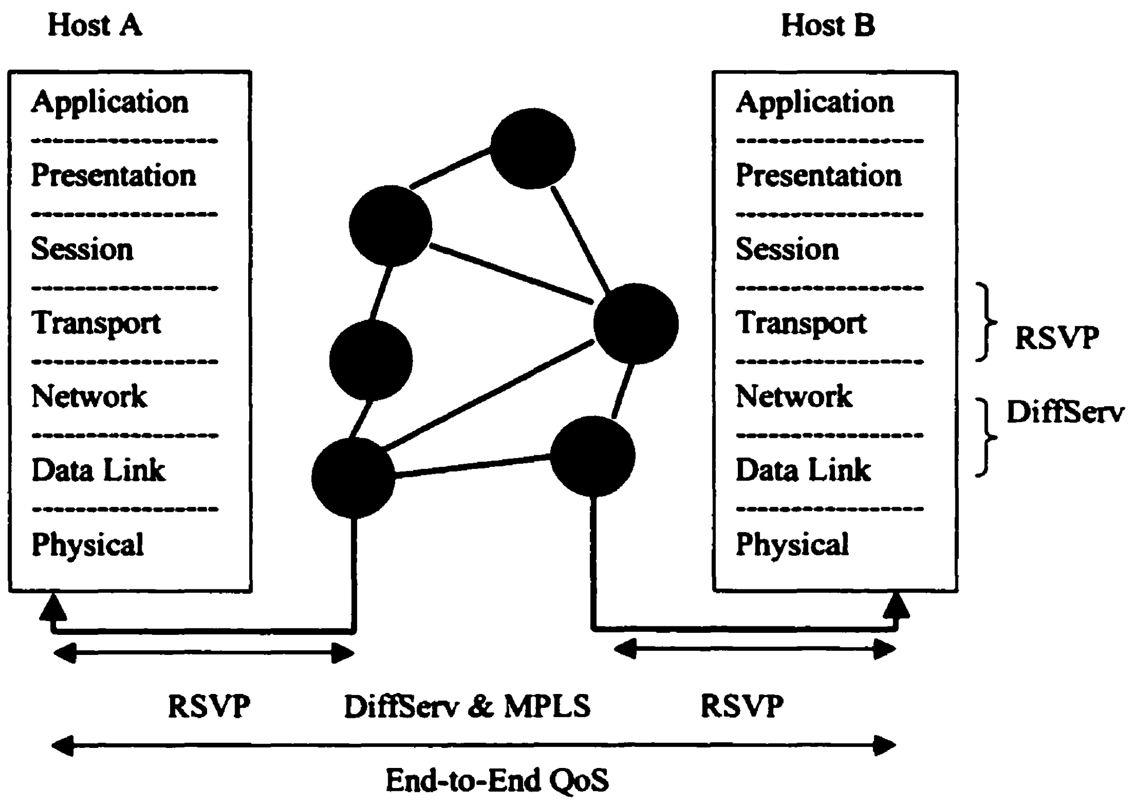


Figure 1.12 End-to-End and Top-to-Bottom QoS

RSVP provisions resources needed for network traffic and DiffServ marks and prioritizes traffic. As already discussed, RSVP is one of the most demanding protocols on backbone routers in terms of complexity and overhead. Therefore, its use should be limited on backbone routers [Man97]. DiffServ can be a complement for RSVP in the quest to ensure End-to-End QoS. In this architecture, the hosts use RSVP to request resource reservation with high granularity. The border routers at the backbone entry points map these RSVP reservations onto a class of service indicated by the DSCP bit pattern. Another alternative is to require the host to set the DSCP bit accordingly also. The border routers at the backbone exit points are responsible for re-establish the RSVP provisioning to the final destination address [Stl99].

1.5 Open Issues

The complexity of distributed networks and the need for consistent management overwhelmed the network operators. The need to capture a higher level of abstraction and to later refine it in order to be implemented raised the question of having an automatic policy management approach. Some achievements presented above partially solved well-defined areas. However, vendors, standard communities and service providers are thriving for defining an uniform approach for policy definition and policy implementation. This will allow proprietary policy-based management tools to interact with each other in a consistent manner. Following this increased need, the IETF proposed a policy framework whose general guidelines are presented in the next subsection.

1.5.1 Implementation QoS Policy Framework

The QoS Policy Schema provides a mapping of QoS policy information to an implementable form in a directory that uses (L)DAP (Lightweight Directory Access Protocol) as its access protocol [Sni00]. QoS policy information includes definition of policy rules, policy conditions, policy actions, and general policy data.

Due to its generality, the QoS Policy schema alone may not be sufficient to model a particular set of QoS services and systems. Three ways in which QoS policy schema may be insufficient exist.

1) The QoS Policy Schema lacks application-specific policies. These extensions can be new functions represented as subclasses of classes defined in the document or as

new attributes of the classes defined in the document. In effect, the QoS policy schema is a middle layer in the following three level hierarchy:

- Core Policy Schema;
- QoS Policy Schema, and
- Implementation-specific schemata.

2) The QoS Policy Schema may not necessarily provide an efficient mapping to a given vendor's directory implementation. Certain LDAP functions are implemented in different ways by different vendors. The basic design may therefore need to be modified in order to fit a particular vendor's implementation.

3) The QoS Policy Schema may lack to accommodate an implementation not compliant with LDAP specifications. This represents a particular case of point 2) above.

Since rules can be as simple as an IF-THEN statement or as complex as an aggregated object, it is necessary to make the representation of management rules uniform.

(a) T. Kock et al [Koc96] presented the general structure of a polling event definition as follows:

```
event name type polling {
  operation (parameter_1, ... , parameter_2)
  every cycle
  [filter ({ median | medium | none }, window) ]
  on { = = | != |  <= |  < | >= |  >} threshold
  [ mode { static | dynamic }]
  [ delay interval ]
  [single]
  trigger eventname
  on ...
 }
```

Name represents the name of the event definition, and *eventname* is the name of the event used to trigger a policy. The *operation* defines an interface method providing the monitoring value. The *cycle* defines the polling rate in seconds. Optionally, a *filter* can be specified together with other technicalities.

The *PolicyRule* class from the IETF DEN (Directory Enabled Networks) is another example of policy rule representation [Str99]. This is a subclass of the *Policy* class and represents the IF [condition is met] THEN [execute action]. This is obtained by aggregating a set of *policyCondition* and a set of *policyAction* objects. A *policyRule* condition is formed by either an ORed set of ANDed conditions (Disjunctive Normal Form) or an ANDed set of ORed conditions (Conjunctive Normal Form). Individual conditions may also be negated. The *policyRule* actions are performed if and only if the *policyRule* is true.

(b) Dini et al. [Din95] present more complex expressions of a management policy in statements of the form:

```
<policy-name> : : = if  {<conditions>, [ |<conditions> or <conditions> |
                        <conditions> and <conditions> ] }
                        then {<actions>, [ |<actions> and <actions> ] }
```

where <conditions> are predicated on values of properties of a system component and <actions> represent management actions. The management actions can be simple updates of component properties or complex management actions performed by managing objects.

We have just seen some proposed policy rule representations, but policy rules must also consider current status of the network elements such as synchronization of some actions or policy triggers, specification of companion policies and so on [Slo94].

1.5.2 Complex filtering, aggregation, correlation mechanisms

Today's telecommunication networks are increasingly complex in their layered structures and services. Integrated backbones using Asynchronous Transfer Mode (ATM) and Synchronous Digital Hierarchy (SDH) are carrying data, voice and video at the same time [Kat97]. The number of protocols in use is therefore high at any given moment. Any problem in the base transmission services causes the failure of numerous higher layer services, which may not even be directly connected to the failing element.

The type of situation mentioned above causes serious problems to the network management systems. If integrated management system is not in place, then the network managers have to manually correlate the problems on the various network management consoles by using their experience. When successful, this procedure takes too long and the system remains down for higher than desired periods of time. If integrated services are present, they become flooded with events from all the different sub-network management systems [Kat93].

Automated fault isolation and event correlation in integrated networks are serious theoretical and practical problems. Fault isolation represents the automated detection of problem, which is the cause of the trouble. When talking about event correlation, all problem indicators whose generation was caused by the same underlying problem are grouped together. The authors of [Kat97] present an algorithm that allows determining the underlying cause of network problems and correlating the events associated with that problem. The main idea is based on model traversing allowing incorporation of different correlation techniques. Model traversing is an approach that reconstructs fault propagation at run-time by analyzing relationships between MOs.

1.5.3 Co-habitation between services

Different perspectives have to be considered when dealing with network management. From the policy perspective, COPS (Common Open Policy Service) and PIBs (Policy Information Base) are promoted. From the management and event triggering perspective, SNMP and MIB are considered by their simplicity and facility to be implemented. At the same time, network resources and services discovery and management require a DEN/LDAP (Directory Enabled Network/Lightweight Directory Access Protocol) schema [How95]. While still in current use, CLI (Command Line Interface) or TL-1 Transaction Language One) must be used with all the above frameworks. It is important to devote appropriate effort to avoid conflicts in applying any or all of these approaches together. All these approaches are considered in further detail below.

(a) SNMP/MIB for network elements

As seen in the previous sections, management is too large and complex that a single vendor cannot handle all its aspects. In order to support the multivendor network management, a standard was needed. The ISO (International Standardization Organization) proposed CMIP (Common Management Information Protocol) while the Internet world inclined towards the SNMP (Simple Network Management Protocol). In 1990, the IETF (Internet Engineering Task Force) published the document that made SNMP an Internet standard. Ever since, it was used in every network management system deployed [Ste00].

SNMP management has the following three components:

1) The Structure of Management Information (SMI)

The SMI defines the structure of the SNMP naming mechanism by identifying the allowable data types for the rules of naming and identifying MIB components. The naming structure is hierarchical and ensures unique names for managed objects, the components of MIB.

SNMP-managed objects are very simple, containing generally six attributes. For example, these attributes can be *name* (ex : ifInErrors), a dotted decimal *object identifier* (1.2.3.4.5.6.7.8.9.10), a *syntax field* that selects the data type (Integer, IPAddress, Counter etc), an *access field* («non-accessible», «read-only», «read-write», «write-only»), a *status field* («mandatory», «optional», «deprecated» or «obsolete») and a *text description field.*

The SMI makes possible the writing an SMI-compliant management object definition, the running of the text through a standard MIB compiler to create an executable and the installing of the code in existing agents and management consoles that could start generating different charts and reports.

2) The Management Information Base (MIB)

The MIB is a hierarchical name space whose nodes registration is administered by the Internet Assigned Numbers Authority (IANA). The term «MIB» can also be attributed to specific collections of objects used for particular purposes. The figure 1.13 below depicts the MIB structure.

The object ID for every relevant object begins with either 1.3.6.1.2.1 (the MIB-2 node) or with 1.3.6.1.4.1 (the Enterprise node). The nodes under MIB-2 include the RMON (remote monitoring) MIB objects and other generic MIB objects such as IP, TCP, UDP etc. The nodes under Enterprise include all proprietary MIB objects. It has been estimated that there are 10 times as many proprietary MIB objects as there are generic ones.

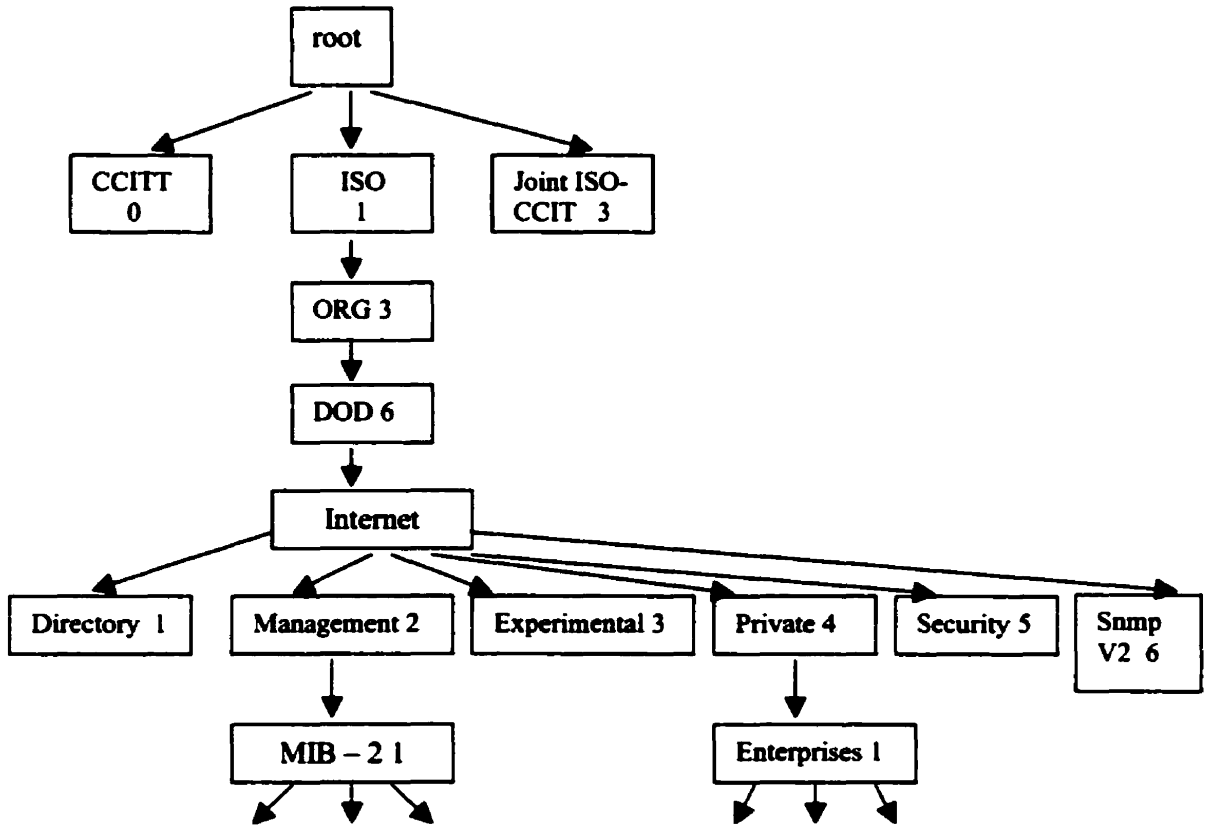


Figure 1.13 The MIB tree of object identifiers

Each MIB object has a value associated with it according to the syntax part of its specification. When a MIB object is instantiated on a device, the value associated with the object is called a MIB variable. SNMP agents store MIB variables and send them upon request to managers on management stations. MIB objects are static. They are compiled from a description language to a binary form that agents and managing objects can load.

MIBs can be divided into smaller units called *groups* of related objects. A vendor can implement the groups that are useful for a product and leave out the ones that are not. For example, MIB-2 has 10 subgroups such as "system", "interfaces", "IP", "ICMP", "TCP", "UDP" and "SNMP". Also, the RMON MIB for Ethernet segments has 9 groups including "statistics", "history", "alarm", "event", "capture". Traditional SNMP agents are not capable of capturing most RMON data; built-in probe functions or special RMON probe devices are necessary to collect and forward RMON information. The advantage of RMON probes extends beyond the capturing and processing of more data than ordinary device agents. In effect, RMON probes can reduce traffic by storing intermediate results locally and forwarding them to applications on demand.

3) Protocol Data Units (PDUs)

The Protocol Data Units represent the different possible payloads that form legitimate management messages.

In SNMP version 1 there are only 5 types of messages:

(a) **get-request** – retrieves one or more values from a managed node's MIB
(b) **get-next-request** – enables the manager to retrieve values sequentially. This operation is widely used when reading through the rows of a table.
(c) **set-request** – enables the manager to update appropriate variables, providing SMNP with the ability to configure and control entities remotely. Managed objects with an access attribute of "read-only" can't be set.
(d) **get-response** – returns the results of «get-request», «get-next-request» and «set-request» operations, at the same time acknowledging them

(e) **trap** – enables an agent to spontaneously report important events or problems to the managing process. Traps are not acknowledged.

Although the first SNMP version was rapidly accepted and implemented, it had serious shortcomings. First, the « password » called community name or community string offered no reliable method of authenticating the source of network management messages. Second, the community name was visible in unencrypted mode in each SNMP packet ; there was no means to secure the contents of network management messages from network hackers. Under such circumstances, many network managers only implemented the SNMP sections responsible for monitoring devices and collecting statistics, hence not taking full advantages of all the SNMP capabilities.

SNMP version 2 addresses the authentication and security problems. It also contains other enhancements such as improved support for systems and applications management, manager-to-manager communication and, hence, a more distributed management model. Retrieving tabular data is more efficient because of the presence of a new message type.

(b) LDAP/DEN for services

Directory Enabled Networks (DEN) provides a schema and information model for representing network elements and services in a directory. An implementation of this specification then enables an appropriate set of network services to be associated with users and applications. The specification defines a set of data models for typical network devices, and implements these as extensions to the directory services schema.

LDAP (Lightweight Directory Access Protocol) is the standard way to access directory information in the Internet. It defines a simplified object model for defining information that can be stored in the directory and gives a corresponding workspace that determines how information is organized, stored and retrieved.

(c) COPS/PIB for polices

COPS is a dedicated protocol for transporting policy information. It is a simple query and response protocol that is used to exchange policy information between a policy server (PDP-Policy Definition Point) and a policy client (PEP-Policy Enforcement Point).

Policy rule classes are arranged in a hierarchical structure similar to the one in SNMP's SMI. For each policy rule class, there may be none or more policy rule instances for any particular device. A collection of policy rule classes are defined in PIB modules, which follow the same structure as MIB [St399].

COPS is the preferred protocol over both SNMP and LDAP for policy configuration installation and communication of policy information. When compared to SNMP, COPS has richer semantics and uses TCP for large transactions (as opposed to SNMP which only uses UDP). When compared to LDAP, COPS has many advantages including (i) carrying and understanding of dynamic state information, (ii) support for unsolicited notifications, and (iii) requirement of less code to be implemented.

(d) Other techniques

The CLI is a simple line-oriented interface used to perform network operations from any node in the network, with or without having a network management system. One limitation of CLI is that it only allows the issue of commands to one node at a time. This means that one cannot view the status of several switches by using a single command, but a separate command for each switch.

Another technique is Transaction Language One (TL-1), a widely used management protocol in telecommunications. It is a man-machine, text-based message set that manages most of the broadband and access networks in North America and is increasingly being used for newer management applications.

1.5.4 Need for mechanisms for early policy conflict detection and resolution

The network heterogeneity, the diversity of services, the diversity of management tools, the complementary protocols and information structures, and the potential conflict between sub-network owners themselves or with their customers, led to a variety of management goals and policies. This variety of management policies, policy conflicts can be of many types. Following is a summary of the most common kinds of policy conflicts:

1) Different policies that refer to the same object may eventually trigger conflicting actions.
2) Different policy owners, having potentially opposite interests, can set different pre-conditions for the same set of actions leading to policy conflicts.
3) The lack of companion policies, that is, policies that have to be in place in order for another policy to be applied, creates policy conflicts.
4) The same policy may trigger conflicting actions : parallel actions, sequential actions or overlapping actions.
5) Different policies may trigger conflicting actions.
6) Different policies can be defined under temporal constraints that must be verified when a policy is really applied. Various kinds of temporal conflicts may occur, as presented in this thesis.

The existence of the above policy conflicts calls for mechanisms for early policy conflict detection and resolution to ensure a successful and effective network management.

To help us understand the current status regarding policies and policy-enabled networks, the next chapter focuses on the current IETF Policy framework and architecture.

Conclusion

Policy-enabled management tools are tailored to cope with the complexity of today's networks and user profiles. Early policy conflict detection and resolution is suitable. Also, temporal aspects of policy definition must present conflicting execution and must favor checking of policy application.

Chapter 2

Current IETF Policy Framework

2.1 Layered Approach

Usually, policies define the essential behavior of distributed heterogeneous systems, applications and networks. The policies only specify the desired behavior, but say nothing about how this behavior can be accomplished and maintained, commonly by using actions pointed by appropriate protocols. Policies range from high level, that is, abstract, non-technical policies to low level, technical policies. The level of abstraction depends on the degree of detail present in the policy definition and the ratio of business issues to technological issues within the policy.

The process of defining, analyzing and structuring policies is the basis for processing and application of management policies, as illustrated in Figure 2.1. Policy classification gives a very simplified representation of the policy, but does not cover all policy aspects. In the *transformation* phase more domains may be resolved. The refinement of the policy classification plus the introduction of policy hierarchy gives a policy template definition [Wei95].

Policies are very different in terms of level of abstraction and degree of detail. A structure of policies is mandatory in order to ensure that all policies can be successfully applied to their targets. Policy hierarchies divide policies into smaller groups of different levels of abstraction, later to be processed into applicable, low-level policies [Mac93][Ngu93]. Weiss presented a policy hierarchy that defines the levels within the management environment in which policies are applied. As depicted in Figure 2.2, this policy hierarchy makes the difference between the following aspects :

- *Corporate or High level policies* – directly following from corporate plans of strategic business management. Technology presence is minimal. In order to be implemented, these policies have to be redefined as one of the other three policy types defined below.

- *Task-oriented policies* – define how management tools must be applied and used in order to achieve the desired behavior
- *Functional Policies* – define the use of management functions, such as the OSI systems management functions [Iso10164], the OSF/DME distributed services [Dme92] and OMG's object services [Omg921, Omg922].
- *Low-level policies* – act at the level of managed objects, the abstractions of managed network and system resources.

In order to be implemented, some policies do not have to arrive at the lowest level in the above hierarchy.

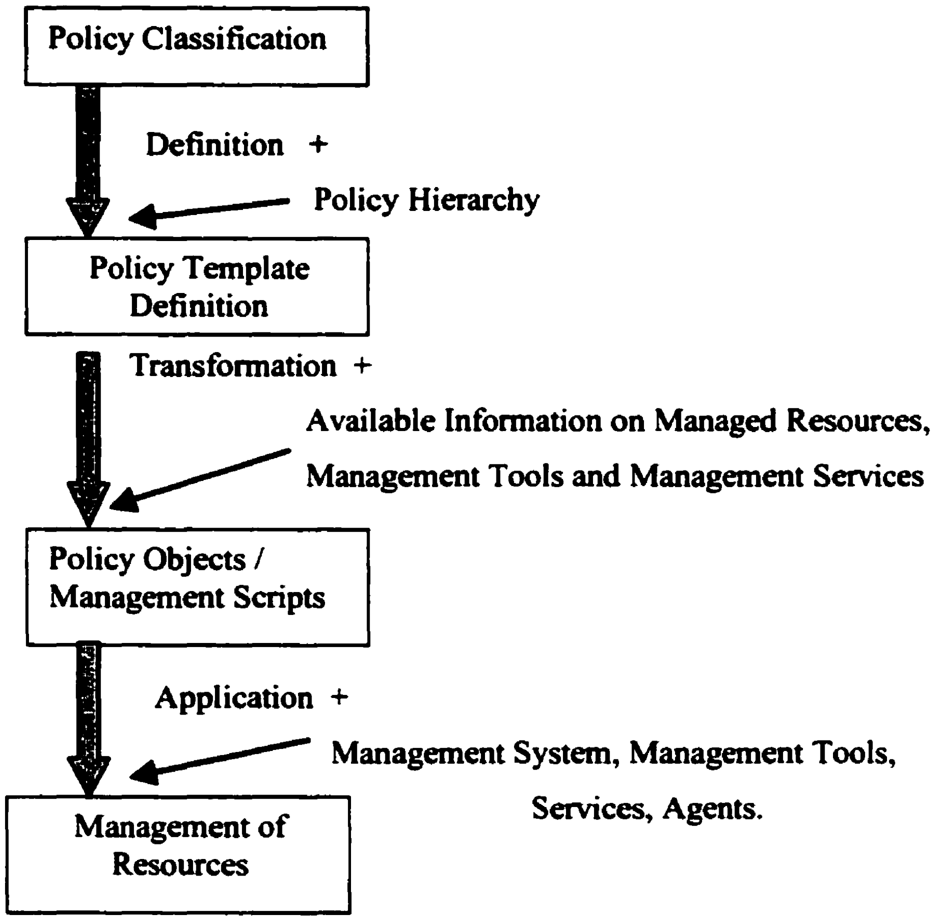

Policy Classification
Definition +
Policy Hierarchy
Policy Template Definition
Transformation +
Available Information on Managed Resources, Management Tools and Management Services
Policy Objects / Management Scripts
Application +
Management System, Management Tools, Services, Agents.
Management of Resources

Figure 2.1 The path from policy classification to policy application

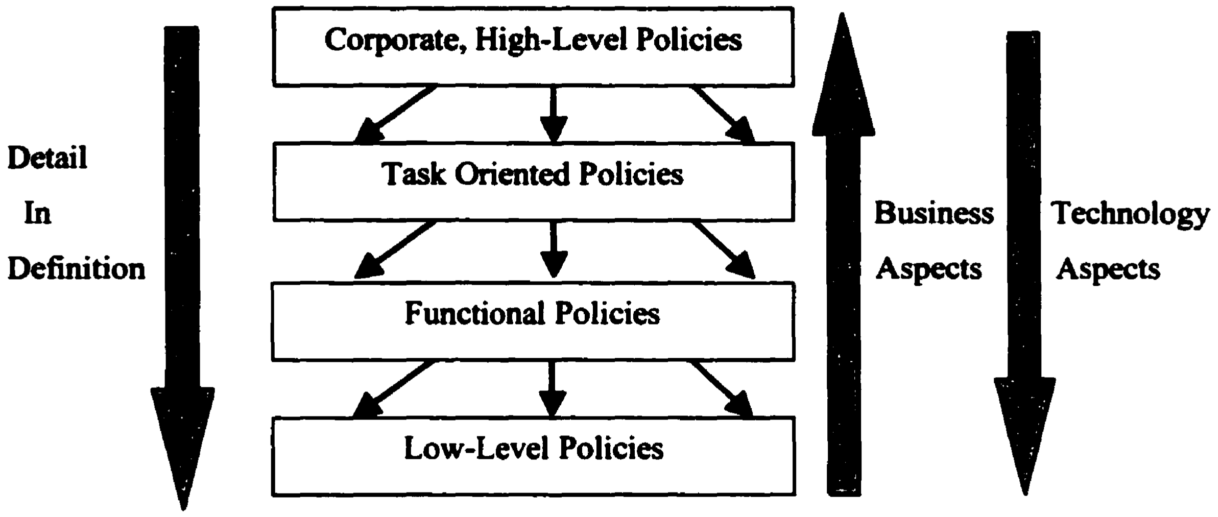


Figure 2.2 A policy hierarchy

The transformation process is used on policies to refine them as well as to identify the targets, subjects and monitor objects. As an example, a high-level security policy can be further refined into two separate policies: one responsible for the access control, the other responsible for the data confidentiality [Iso7498].

In some situations, the refinement process may be automated entirely, but, usually, it is only partially automated. Extensive information on the managed environment, the managed capabilities, the tools and platforms available is essential for the interpretation of the policy semantics. An asyntactical approach to refinement is not desired since it may cause most of policy conflicts. A refinement is considered complete when the lowest level of detail has been reached or when a mapping between the value to managed objects or management functions of the management system is possible.

Another policy hierarchy has been presented by Strassner [Str99]. Let us consider the DEN (Directory Enabled Networks) policy class hierarchy in Figure 2.3 below.

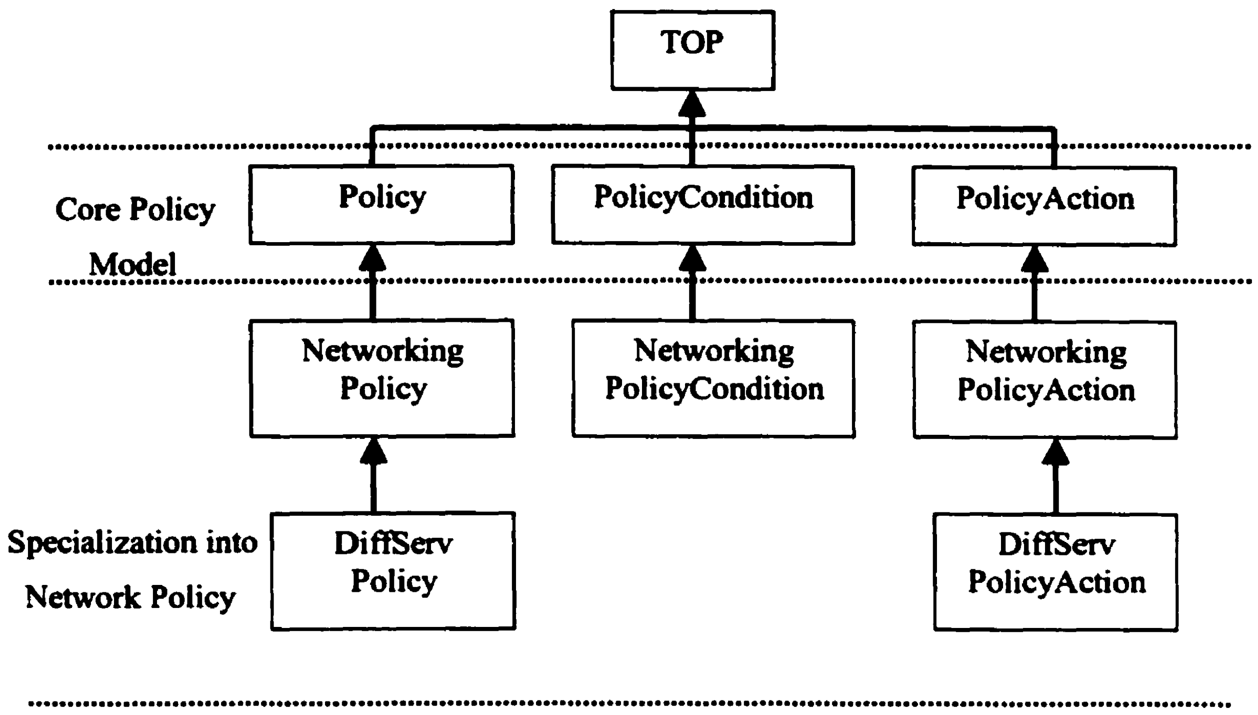


Figure 2.3 The DEN policy class hierarchy

The *Policy Class* is a subclass of the *TOP* class and is used to contain information on the use and interaction between network resources and services in a specific context. A policy represents an aggregation of one or more *PolicyCondition* and one or more *PolicyAction* objects.The *NerworkingPolicy* class is a subclass of the *Policy* class and is a base class for grouping different networking policies. The *DiffServ Policy* class is a subclass of the *NerworkingPolicy* class and is used to define policies that apply specifically to using DiffServ as a tool of achieving the required results. The *PolicyCondition* class is a subclass of *TOP* and describes a set of conditions necessary to determine if the actions associated with a certain *Policy* should be executed. The *NetworkPolicyCondition* class is a subclass if *PolicyCondition* class and provides a canonical representation of network policy. The *PolicyAction* class is a subclass of TOP and describes a set of actions that are invoked when the conditions for a *Policy* are satisfied. The *NetworkingPolicyAction* class is a subclass of the *PolicyAction* class and provides a canonical representation of common network policy actions. The

DiffServPolicyAction class is a subclass of the *NetworkingPolicyAction* class and is used to describe actions specific to the IETF DiffServ effort [Str99].

2.2 Policy Framework and Architecture

2.2.1 Foundation

Many organizations have policies concerning security or specifications of rights and responsibilities related to a particular executive position. A role framework has been developed [Lup297], which can be used to formally express role concepts (explained further), analyze these specifications for consistency and translate them into automated agents to manage distributed systems. The *relationships* between multiple roles can also be considered to define the rights, duties and protocols related to interactions between roles.

Policies

A *domain* is a collection of references to object interfaces that have been grouped in order to facilitate their management. The concept is very similar to a directory in a hierarchical file system. A *domain service* manipulates the membership information. Also, *domain scope expressions* can be specified and represent the set of objects to which a policy applies. Policies that apply to a domain, will, by default, apply to all the sub-domains of that domain, but this default can optionally be turned-off. A *user representation domain (URD)* is a persistent representation of the human in the computer system. When a user logs onto the system, an adapter object is created within the URD and acts as the interface between the person and the computer system. Other agents representing the human can also be created in the URD.

Management *policies* are rules used to change the behavior of a system. They establish relationships between managers and managed object domains, which can be either Obligation or Authorization. "Rights" can be modeled as *authorization policies* specifying what activities a user, manager or agent is permitted or forbidden to perform on a set of objects. "Duties" can be modeled as *obligation policies* specifying what activities a user, manager or agent must or must not perform on a set of objects. By separating policies from managers that interpret them allows for the change of the behavior and strategy of the management system without re-coding the managers.

The general format of a policy is [Lup97b]:
Identifier Mode (Trigger) Subject '{'Action'}' Target (Constraint) (Exception) (Parent) (Child) (Xref) ';'

The different *modes* possible are: (permitted = A+) (forbidden = A-) (must = O+) (must not = O-). The *subject* represents the set of managers designed to carry out the actions. The constraints limit the applicability of the policy.

Policies can specify actions at different levels of abstraction. Authorization policies are translated into access control lists to be interpreted by security agents in the target system, while obligation policies given to distributed automated management systems for interpretation.

There are two situations in which several policies may apply to the same objects: (1) when objects are members of the same domain and (2) when different management functionalities need to be reflected by the same object.

Roles

A role represents a group of policies specifying obligations and authorizations, and a relationship represents a group of interacting roles [Lup97a]. A role groups the policies specifying the duties and rights of a particular position inside an organization.

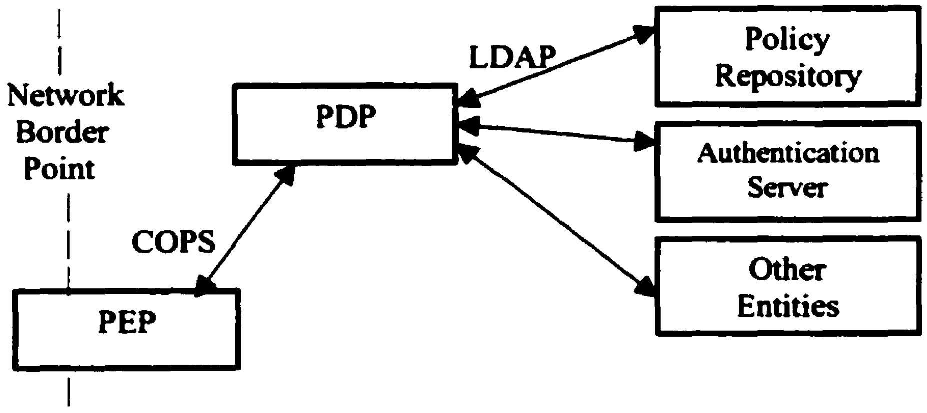
Policies reference a common subject domain called the Manager Position Domain (MPD). The implementation of a role is an object which maintains a reference to the MPD of the role and a table of all the policies which are part of the role. Each entry in this table contains a unique name within the role, and a reference to the policy object to be resolved by the underlying support system.

The issue of concurrency constraint rules, which apply to policies of a role or of a relationship, has been addressed [Lup97b]. It allows the expression of sequences of actions, parallelism and synchronization. Concurrency constraints can be translated to complex events that will trigger the execution of activities. A role object maintains a concurrency constraint table containing a unique name and a constraint expression.

2.2.2 Policy framework

In order to establish a scalable policy control model for the ReSerVation Protocol (RSVP) and the integrated services (IntServ) it enables, the RSVP Admission Policy working group in the IETF (Internet Engineering Task Force) was created. The group proposed a policy framework that can be applicable to other QoS technologies, such as differentiated services (DiffServ). The framework is also useful for other technologies that need policy support such as network security (firewalls, system access, IP security, VPN (Virtual Private NetworkS), etc) [Sta99].

Two main components are at the core of the policy framework: the Policy Decision Point (PDP) and the Policy Enforcement Point (PEP). PDP is the decision-maker unit and PEP is the enforcer unit. The proposed policy framework is depicted in Figure 2.4 below:



Legend: PEP=Policy Enforcement Point; COPS=Common Open Policy Service;
PDP=Policy Decision Point; LDAP=Lightweight Directory Access Protocol

Figure 2.4 The Policy Framework

The PEP sends policy elements with the request and receives in turn a decision from PDP, possibly with policy elements such as errors. PDP exports information from monitoring and management units through Simple Network Management Protocol (SNMP), Lightweight Directory Access Protocol (LDAP) etc. Here, the PDP and the PEP are considered separate entities from a functional point of view, but, physically, they can be built into the same device. As a matter of fact, the Policy Framework specification [Yav99] describes a sub-component of the PEP called the Local PDP (LPDP) that enables PEP to perform some decisional functions. However, a PEP is always required to send a request to the PDP for final policy decision.

A policy-enabled network is likely to have many PEPs in any network domain with multiple interfaces on each PEP. On the other hand, the number of PDPs or Policies Repositories is likely to be much lower in order to simplify administration. For this reason, they will be more centralized than PEPs, although still distributed. Having more than one PDP or Policy Repository per network domain will provide some fault tolerance [Sta99].

2.2.3 Policy functions

In addition to the functions of *decision* and *enforcement* that are assigned to PDP and PEP respectively, *metering* is another important function. The following is a brief description of these three primary policy functions:

Decision-making: Decision-making represents the function of PDP and involves retrieving and interpreting policies, detecting policy conflicts, receiving interface (Role) descriptions, policy decision requests and policy elements (Conditions) from PEPs, determining which policy is relevant, applying the policy and returning the results. Decision-making by PDP uses static or dynamic data in order to determine if a policy is being satisfied and, if not, what steps need to be taken in order to satisfy it. Decision-making also involves sending policy elements to the PEP "asynchronously," based on policy updates or requests from external entities. The COPS "Policy Provisioning Client" message [Rei99] was created for this purpose, and it sends a Basic Encoding Rules-encoded policy instance called a "PIB" to be installed in the PEP.

Enforcement: Enforcement represents the function of PEP and involves PEP taking actions according to the PDP decisions and based on relevant policies and current network conditions. The network conditions, in turn, can be static such as source and destination addresses or they can be dynamic such as current bandwidth availability and time of day.

Metering: Metering represents the active or passive examination of the network and its constituent devices. The elements recorded and verified are network health, whether policies are being satisfied and whether the clients are taking unfair advantages over the network services. Metering by PEP is the auditing of policy compliance to verify that the policy consumers properly implement policies.

2.2.4 Policy architecture

The policy architecture describes how a policy management tool is related to the other policy framework components.

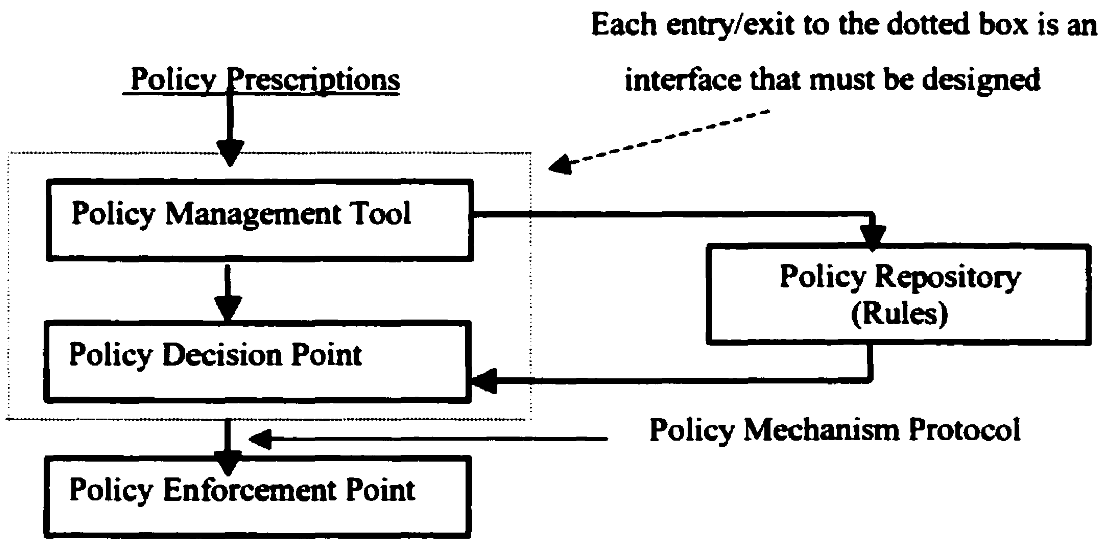

Each entry/exit to the dotted box is an interface that must be designed
Policy Prescriptions
Policy Management Tool
Policy Repository (Rules)
Policy Decision Point
Policy Mechanism Protocol
Policy Enforcement Point

Figure 2.5 The Policy Architecture

2.2.5 Policy Core Information Model

The Policy Core Information Model (PCIM) defines the structure of generic policy information [Str00]. It defines two hierarchies of object classes: structural classes representing policy information and control, and association classes defining how instances of the structural classes are related to each other [Sni00b]. It represents objects and relationships between objects, and, in this sense, it is not confined to a specific repository. The information model defines the mappings that translate the data specified in the information model to a specific type of repository.

The Policy Core Schema defines a mapping that interprets the information from the PCIM in a form that can be implemented in a directory. PCIM extends as to include information needed to represent QoS policies with the QoS Information Model [Sni00a]. The mapping of these information model classes to a directory that uses LDAPv3 as its access protocol are defined in [Sni00b]. For the classes in the information model, the mapping is done one-to-one, that is, information model classes map to LDAP classes and information model properties map to LDAP attributes.

The policy core schema consists of thirteen classes. From these, six are general classes, namely *policy*, *policyGroup*, *policyRule*, *policyCondition*, *policyTimePeriodCondition*, *policyAction*, and two are vendor-specific classes, namely *vendorPolicyCondition* and *vendorPolicyAction* [Wah00]. The Core Model only presents general policy classes, but application-specific policy models can be derived from this general model by using the *PolicyGroup*, *PolicyRule*, and *PolicyTimePeriodCondition* classes.

```
TOP (root of the directory)
 |
 |------ policy (abstract)
 |          |------------ policyGroup (structural)
 |          |----------- policyRule (structural)
 |          |----------- policyCondition (auxiliary)
 |          |                   |----------- policyTimePeriodCondition (auxiliary)
 |          |                   |----------- vendorPolicyCondition (auxiliary)
 |          |
 |          |----------- policyAction (auxiliary)
 |          |                   |----------- vendorPolicyAction (auxiliary)
 |          |
 |          |----------- policyInstance (structural)
 |          |----------- policyElement (auxiliary)
 |
 |------ policySubtreePtrAuxClass (auxiliary)
 |------ policyGroupContainmentAuxClass (auxiliary)
 |------ policyRuleContainmentAuxClass (auxiliary)
```

Figure 2.6 Class hierarchy in policy core schema

The schema also contains two auxiliary classes *policyGroupContainmentAuxClass* and *policyRuleContainmentAuxClass* and two other classes defined for optimized LDAP retrievals : *policySubtreesPtrAuxClass* and *policyElement*. The last class, *policyInstance*, is defined for attaching auxiliary classes representing policy conditions and policy actions [Sta99]. The Figure 2.6 shows schematically the classes hierarchy. Section 2.2.6 will present more details on these classes as well as on they are used to describe policies. Subclasses and auxiliary classes extend *PolicyCondition* and *policyAction*.

2.2.6 Policy condition and policy action classes

We have seen that policy core schema defines general classes for policies. However, policy applications requiring QoS need specialized subclasses derived from *policyCondition*.

The *networkingPolicyConditions* subclass has five auxiliary subclasses (Figure 2.7, adapted from [Sta99]) used by network managers to control access to network resources and services [Raj99a]:

- Host – host ID or source and/or destination address;
- User – user identifier type and values for senders and receivers;
- Application – source and/or destination port number ranges, transport protocols and/or received IP ToS byte values;
- Routing – Interfaces by Ipv4 or Ipv6 address or interface ID and the traffic direction in them, and
- Layer 2 – source and/or destination MAC (Media Access Control) address range, Ethertype, 802.1Q VLAN (Virtual Large Area Network) identifier, etc.

TOP (root of the directory)

```
   |
   |------ policyCondition (structural)
   |            |-------- networkingPolicyCondition (structural) ____
   |                                                                 |
   |---------------------- hostConditionAuxClass (auxiliary) _____ |
   |---------------------- userConditionAuxClass (auxiliary)
   |---------------------- applicationConditionAuxClass (auxiliary)
   |---------------------- routeConditionAuxClass (auxiliary)
   |---------------------- layer2ConditionAuxClass (auxiliary)
```

Figure 2.7 Class hierarchy for the *networkingPolicyCondition* subclass and other policy condition-related auxiliary classes

A vendor can attach any or all of these auxiliary classes to *networkPolicyConditions* class in order to meet his/her needs. Their definition simplifies management, assuring extensibility and reusability. Different vendor goals are achieved by extending *networkPolicyConditions* class differently. Also, the auxiliary classes may be associated with other subclasses of *policyCondition*, *DHCPPolicyCondition* etc.

Since LDAP does not support multiple inheritance, a new subclass has to be created each time new attributes need to be added, starting from structural objects. For the auxiliary classes, the attributes can be mixed into an instance without attribute clashes. Because of this, all attributes should be optional in auxiliary classes, but some should be required [Sta99].

Similar to the *policyCondition* class, the *policyAction* class also contains subclasses responsible for QoS facilitation [Raj99b]. Depicted in Figure 2.8, these policy action subclasses support RSVP and Differentiated Services. The idea was to provide support QoS policies for the following situations:

- the use of RSVP as a signal mechanism for integrated services inside a domain or between domains (per flow reservations);
- the use of DiffServ inside a domain or between domains (prioritized aggregates), and
- the mapping of RSVP from one domain onto DiffServ on another domain (mapping per-flow reservations onto aggregates at a domain entry point and back at a domain exit point).

TOP (root of the directory)

```
   |
   |------ policyAction (structural)
   |            |-------- diffServAction (structural)
   |            |-------- rsvpAction (structural)
   |---------------------- diffServResourceGroup (structural)
   |---------------------- RSVPResourceGroup (structural)
```

Figure 2.8 Class hierarchy for the policy action subclasses and related group classes

2.3 COPS and PIBs

The following sections describe in more detail the issues related to COPS and PIBs that have been introduced in Chapter 1.

2.3.1 Policy Information base

Policy-enabled networks help network managers to allocate network resources more effectively and reduce the management overhead. These networks require policy-aware servers, switches, routers and end-stations. Policy servers, also known as Policy Decision Points (PDP), are designed to make most policy decisions. Policy-aware routers and other network elements, known as Policy Enforcement Points (PEP) are designed to control the access to network resources.

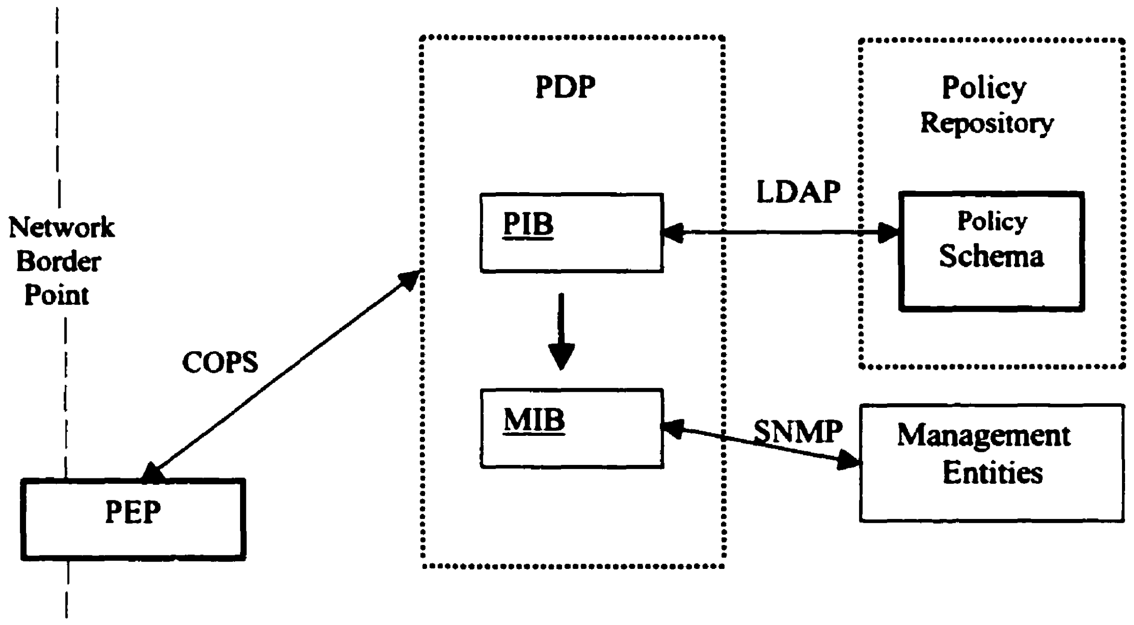


Figure 2.9 The Policy Information Base (PIB)

Policy is represented and stored in a policy repository from where it is retrieved by PDP using LDAPv3. Aside from the declarative information of policy rules, other vendor-specific information is needed. To interface the protocol-specific information, a Policy Information Base (PIB) was introduced, as depicted in Figure 2.9. As mentioned in

Chapter 1, Section 1.5.4, PIB modules hold the collection of policy rule classes and follow the same structure as MIBs for SNMP.

2.3.2 COPS (Common Open Policy Service) protocol

In order to allow PDP and PEP to interoperate, the protocol Common Open Policy Service (COPS) is used. COPS is based on the query/response paradigm and needs TCP/IP for its transport mechanism. Each COPS message carries client specific-information together with the actions the requestor asks from a device. COPS protocol allows for ten different message types, such as Request, Response, Report State, Synchronize State, Client-Accept, Client-Close, Keep-Alive [Coo00].

2.3.3 LDAP Policy Schema

The LDAP implementation of the CIM policy model forms the basis for the DEN Policy Model. Some differences exist between the IETF Policy Model and the LDAP corresponding implementation [Str99]. These differences are due to the limitations of LDAP and to take into account how information is organized in a directory. This organization is different from other types of repositories, so a specialized mapping is needed.

2.4 QoS Policy Schema

The QoS policy schema has at its basis the object-oriented QoS policy information model which is presently developed by the IETF Policy Framework Working Group. The QoS policy information model defines the structural and auxiliary object classes used for QoS policy information representation. These classes extend the Core Policy object classes, as seen in the Policy Core Information Model in Section 2.2.5. More precisely, the Policy Core Information Model defines the generic structure of a policy and describes a framework for declaring specific conditions and actions used to built application and domain-specific policies. The QoS Policy Information Model refines the Policy Core

Information Model to include policy rules, conditions, and actions needed for QoS network policies representation. Some of these concepts are discussed in the following.

2.4.1 Mappings between Policy Core Information schema and QoS information model

Since information models are repository-independent, there is a need to map the data contained in the information model to a form that can be implemented in a specific repository.

In order to ensure this, two mappings are defined by the Policy Framework Working Group. The first mapping is the Policy Core Schema, which maps the information in the Policy Core Information Model to a form that can be implemented in a directory. The second mapping is one from the QoS Information Model to a form suitable to be implemented in a directory.

2.4.2 QoS policy information model

The QoS policy information model provides the detailed semantics for the QoS policy classes. It refines the Core policy information model by refining the concept of generic policy rules, conditions and actions to cover extensions necessary for representing QoS policies. The QoS policy information model also introduces QoS capabilities to integrated and differentiated services by allowing policy control on RSVP admissions [Sni00a]. The classes defined here are mapped onto a directory that uses LDAPv3 as its access protocol, as will be described in Section 2.4.3.

Different services have different capabilities so they may respond differently to the same high-level policy rule. To avoid this problem, a set of common abstractions is defined to be used to build high-level QoS policies. Different devices use the same low-level abstractions of mechanisms when implementing QoS services. Also, different policy servers and applications provision parts of the network differently if there is a lack of a common high-level policy.

Figure 2.10 depicts the relation between classes in Policy Core information model and classes in the QoS policy information model. Each class is clearly identified as belonging to one of these models.

```
[unrooted]
 |
 +--Policy (Core model)
 | |
 | +---PolicyGroup (Core model)
 | |     |
 | |     +---qosPolicyDomain (QoS model)
 | |     |
 | |     +---qosNamedPolicyContainer (QoS model)
 | |
 | +---PolicyRule (Core model)
 | |
 | +---PolicyCondition (Core model)
 | |     |
 | |     +---PolicyTimePeriodCondition (Core model)
 | |     |
 | |     +---VendorPolicyCondition (Core model)
 | |     |
 | |     +---qosPolicySimpleCondition (QoS model)
 | |
 | +---PolicyAction (Core model)
 | |     |
 | |     +---VendorPolicyAction (Core model)
 | |     |
 | |     +---qosPolicyPRAction (QoS model)
 | |     |
 | |     +---qosPolicyRSVPAction (QoS model)
 | |     |
 | |     +---qosPolicyRSVPSignalCtrlAction (QoS model)
 | |     |
 | |     +---qosPolicyRSVPInstallAction (QoS model)
 | |     |
 | +---qosPolicyPRTrfcProf (QoS model)
 | |
 | +---qosPolicyRSVPTrfcProf (QoS model)
 | |
 | +---qosPolicyVariable (QoS model)
 | |
 | +---qosPolicyValue (QoS model)
 | |     |
 | |     +---qosPolicyIPv4AddrValue (QoS model)
 | |     |
 | |     +---qosPolicyIPv6AddrValue (QoS model)
 | |     |
 | |     +---qosPolicyMACAddrValue (QoS model)
 | |     |
 | |     +---qosPolicyStringValue (QoS model)
 | |     |
| |   +—qosPolicyBitStringValue (QoS model)
| |   |
| |   +—qosPolicyDNValue (QoS model)
| |   |
| |   +—qosPolicyAttributeValue (QoS model)
| |   |
| |   +—qosPolicyIntegerValue (QoS model)
| |
| +—qosPolicyMeter
| |
| +—qosPolicyPHBSet (QoS model)
| |
| +—qosPolicyPHB (QoS model)
| |
+--CIM_ManagedSystemElement (abstract, Core model)
    |
    +--CIM_LogicalElement (abstract, Core model)
      |
      +--CIM_System (abstract, Core model)
        |
        +—CIM_AdminDomain (abstract, Core model)
            |
            +—PolicyRepository (Core model)
```

Figure 2.10 Hierarchy relation between Core and QoS policy classes

The class *qosPolicyDomain* defines the root of a single administrative QoS policy domain, and contains the domain's policy rules and definitions. The class *qosNamedPolicyContainer* represents an administratively defined policy rule container. All policies that are commonly administered are defined in a particular *qosNamedPolicyContainer*. This allows the administrator to group different sets of policy rules that perform different types of operations. The class *qosPolicyPRAction* defines DiffServ actions to be applied on a flow or group of flows, including the marking of a DSCP value, dropping, policing and shaping. The class *qosPolicyRSVPAction* defines a policy action to be applied on an RSVP signaling message that matches the rule condition. The class *qosPolicyPRTrfcProf* is a class that carries the policer or shaper rate values to be enforced on a flow or a set of flows. The class *qosPolicyRSVPTrfcProf* represents an IntServ RSVP Traffic profile. Values of RSVP traffic profiles are compared against Traffic specification and QoS Reservation requests from the RSVP requests. The class *qosPolicyRSVPSignalCtrlAction* extends the functionality of the *qosPolicyRSVPAction* class by adding detailed control on the signaling protocol behavior itself. The class *qosPolicyRSVPInstallAction* extends the functionality of the qosPolicyRSVPAction class by adding detailed control for COPS Install decisions. The auxiliary class *qosPolicySimpleCondition* allows the declaration of simple conditions composed of an ordered triplet: <Variable> <Operator> <Value>. If <variable> matches <value>, the condition is evaluated to True. With the class *qosPolicyVariable*, variables are used to build individual conditions; variables specify the properties of a flow and should be matched when evaluating the condition. The class *qosPolicyValue* is used for defining values and constants used in policy conditions. The class *qosPolicyIPv4AddrValue* is used to provide a list of IPv4Addresses, hostnames and address range values and the class *qosPolicyIPv6AddrValue* is used to define a list of IPv6 addresses, hostnames, and address range values. The class *qosPolicyMACAddrValue* defines a list of MAC addresses and MAC address range values. The class *qosPolicyStringValue* is used to represent a single or set of string values, whereas the class *qosPolicyBitStringValue* is used to represent a single or set of bit string values. The class *qosPolicyDNValue* represents a single or set of Distinguished Name (name that can be used as a key to retrieve an object) values, including wildcards. The class *qosPolicyAttributeValue* is used to represent a single or set of property values in an object. The class *qosPolicyIntegerValue* provides a list of integer and integer range values. The class *qosPolicyPHBSet* serves as a named container for *qosPolicyPHB* objects. The abstract class qosPolicyPHB extends the policy class with the information required to model a PHB service class. The class *qosPolicyMeter* models a meter, that is, a way to measure the temporal properties of the stream of packets selected by a classifier against a traffic profile.

2.4.3 Inheritance hierarchy for the LDAP QoS Policy Schema

Figure 2.11 depicts the class hierarchy for the LDAP QoS Policy schema (adapted from [Sni00b]). The classes that are found in the Policy Core schema are presented in bold.

```
TOP
|
+--policy (abstract)
|  |
|  +--policyGroup (structural)
|  |  |
|  |  +--qosPolicyDomain (structural)
|  |  |
|  |  +--qosNamedPolicyContainer (structural)
|  |
|  +--policyRule (structural)
|  |
|  +--policyRuleConditionAssociation (structural)
|  |
|  +--policyRuleActionAssociation (structural)
|  |
|  +--policyInstance (structural)
|  |
|  +--policyConditionInstance (structural)
|  |
|  +--policyActionInstance (structural)
|  |
|  +--policyElementAuxClass (auxiliary)
|  |
|  +--policyConditionAuxClass (auxiliary)
|  |   |
|  |   +--qosPolicySimpleCondition (auxiliary)
|  |
|  +-- qosPolicyMeter (auxiliary)
|  |
|  +-- qosPolicyPRTrfcProf (auxiliary)
|  |
|  +-- qosPolicyRSVPTrfcProf (auxiliary)
|  |
|  +-- qosPolicyPHBSet (abstract)
|  |
|  +-- qosPHB (abstract)
|  |
|  +--qosPolicyVariable(auxiliary)
|  |
|  +--qosPolicyValue(abstract)
|  |   |
|  |   +--qosPolicyIPv4AddrValue(auxiliary)
|  |   |
|  |   +--qosPolicyIPv6AddrValue(auxiliary)
|  |   |
|  |   +--qosPolicyMACAddrValue(auxiliary)
|  |   |
|  |   +--qosPolicyStringValue(auxiliary)
|  |   |
|  |   +--qosPolicyBitStringValue(auxiliary)
|  |   |
|  |   +--qosPolicyDNValue(auxiliary)
|  |   |
|  |   +--qosPolicyAttributeValue(auxiliary)
|  |  |
|  |  +--qosPolicyIntegerValue(auxiliary)
|  |
|  +--policyActionAuxClass (auxiliary)
|        |
|        +-- qosPolicyPRAction (auxiliary)
|        |
|        +-- qosPolicyRSVPAction (auxiliary)
|        |
|        +-- qosPolicyRSVPSignalCtrlAction (auxiliary)
|        |
|        +-- qosPolicyRSVPInstallAction (auxiliary)
|
|
+--policyRepository (structural)
|
+--policyGroupContainmentAuxClass (auxiliary)
|
+--policyRuleContainmentAuxClass (auxiliary)
```

Figure 2.11 QoS Policy Schema Inheritance Hierarchy

The classes defined in the Policy Core Information Model were dealt with in Section 2.2.5 and the classes defined in the QoS Policy Information Model were presented in Section 2.4.2. To achieve the mapping of the information model's relationships, the schema contains two auxiliary classes: *policyGroupContainmentAuxClass* and *policyRuleContainmentAuxClass* [Str00b]. The LDAP policy schema mentioned in Section 2.3.3 is to be used by implementations that use LDAP as their policy repository.

2.4.4 Containment and Scoping Policy

Containment is an important characteristic of directories. Figure 2.7 depicts the general view of the QoS policy classes containment.

In Figure 2.12, *implied containment* means that the class *policyGroup* would not contain an instance of the *policyRepository* class, but would contain instances of the classes contained in the *policyRepository* class.

The QoS policy hierarchy is a set of containment relationships. This hierarchy is composed of container objects that each have sets of Domain Names that point to other objects from the *policyRepository* class. These other objects from the *policyRepository* class model containment by placing the contained objects as leaf nodes of the container. The container objects from the *policyGroup* model containment both by placement and Domain Name reference.

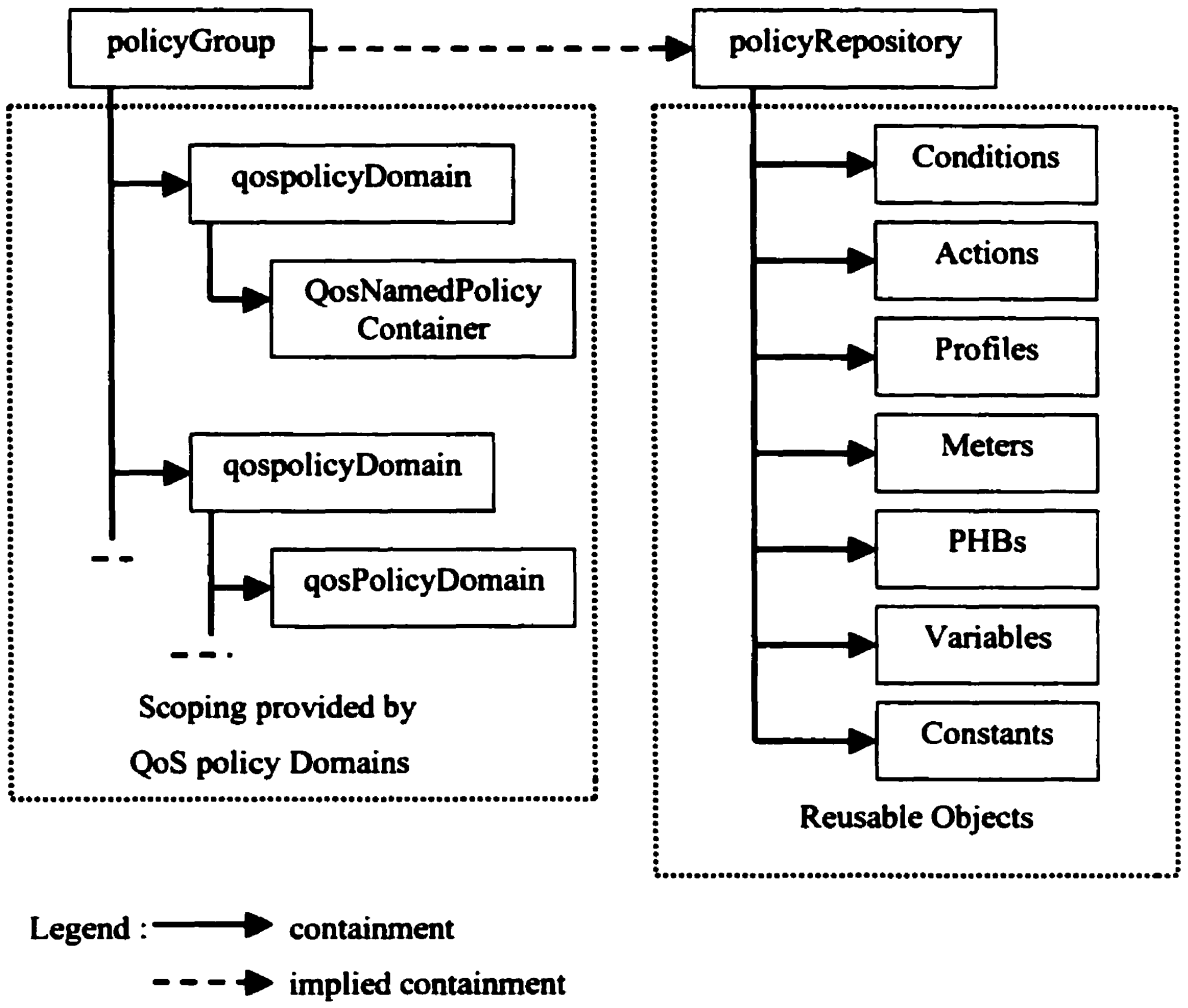


Figure 2.12 QoS policy class containment

In terms of the Policy Framework Core Schema, containers are based on auxiliary classes. A container may be attached to a branch-level class so that specific leaf-nodes of sub-domains, applications etc may be added below the branch-level class. Policy scoping is depicted in Figure 2.13.

There are two types of objects:

- *Reusable Objects* – these are objects that can be used by many policy rules. They reference cross the containment hierarchy and are not considered part of the policy tree structure. The advantage they bring is that one object can be used by many policy rules. This allows one set of conditions and actions to be defined once and used many times. One drawback is that in the directory implementation, the object may demand multiple accesses, which can become costly.
- *Rule-Specific Objects* – these are objects that can be used only by a single rule. Their advantage is the speed of access and a simplified design. The drawback is that there is no reusability and inconsistencies may arise if the same entity is defined in different ways when used by multiple rule-specific objects [Sni00b].

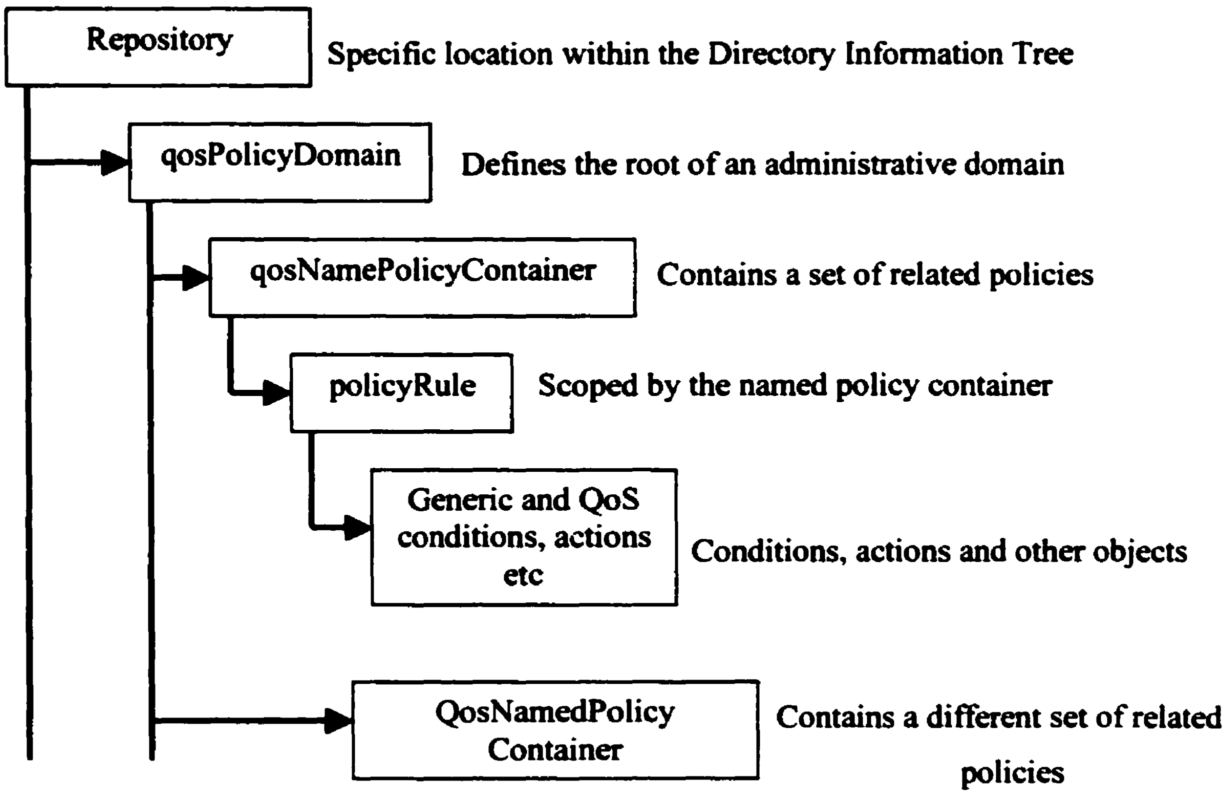

Repository
Specific location within the Directory Information Tree
qosPolicyDomain
Defines the root of an administrative domain
qosNamePolicyContainer
Contains a set of related policies
policyRule
Scoped by the named policy container
Generic and QoS conditions, actions etc
Conditions, actions and other objects
QosNamedPolicy Container
Contains a different set of related policies

Figure 2.13 Containment Hierarchy and Policy Scoping

2.4.5 Features of QoS Policy rules

Section 2.4.2 mentioned the policy QoS class *qosNamedPolicyContainer* which represented an administratively-defined policy rule container. Each *qosNamedPolicyContainer* contains a set of policy rules or other policy containers. QoS policy rules are modeled by the*policyRule* class seen in the Policy Information Core Model.

A rule is a statement of the form "IF condition THEN action". The application of a rule implies evaluating its condition and execute the action when it evaluates to TRUE or do nothing when it evaluates to FALSE. A policy rule must belong to only one *qosNamedPolicyContainer* to retain its coherence. Should it belong to more than one container, many important rule features would be lost. For example, a rule would lose its ability to know its position inside a container, a property based on the relative values of the Priority attribute of each of the*policyRules*, because it would belong to many containers at the same time. Rule ordering would then be impossible.

A rule is a composite object that contains conditions and actions. A condition is a Boolean expression that is evaluated to see if the rule should apply. When it evaluates to true, the corresponding actions should be taken. Actions are specifications of QoS operations.

The basic IF-THEN policy rule is extended to include other rule characteristics that are briefly described below.

Policy Rule Structure

In addition to the basic rule attributes, the following QoS-related attributes are defined :

- *Enable flag* indicating if a rule is enabled, disabled or enabled in debug mode
- *Condition set* defined in *PolicyConditionInPolicyRule* or *PolicyConditionInPolicyRepository* aggregation
- *Flag* indicating if the condition is conjunctive or disjunctive normal form

- *A list of actions* defined in *PolicyConditionInPolicyRule* or *PolicyConditionInPolicyRepository* aggregation
- *Priority value* inside a container
- *A " mandatory" attribute* used to define whether the conditions evaluation and execution is mandatory or not
- *A SequenceActions* attribute that defines how to execute the actions if the condition is true
- An array of *policyRoles* attributes defining the roles used in the rule
- A *RuleUsage* attribute describing how the rule should be used.

So, if the rule is enabled and the conditions are evaluated to true, then the list of actions should be used to dictate how this flow should be treated.

QoS policy conditions

A condition is represented as either an ORed set of ANDed conditions or an ANDed set of ORed conditions. Every single condition may be negated. From the networking point of view, some conditions may be modeled as filters. There is no filter class defined, but it is modeled in the Network Model of DEN.

The *qosPolicySimpleCondition* class models individual conditions. This class refines the *policyCondition* class and uses the triplet (<variable>, <operator>, <value>) to form a condition. The <variable> field specifies the attribute of a flow that should be matched when the condition is evaluated. Simple conditions may be reused by placing them in a common portion of the policy information tree. A simple condition may be added to a policyRule in two ways:

- by direct attachment of an instance of the condition to the ConditionInPolicyRule instance in what is called an ad-hoc simple condition. This method creates private (not reusable) simple conditions.
- by indirect reference to an instance of a condition from a reusable-object repository.

Four different ways to compose simple conditions are:

- Variable Representation
  - the class *qosPolicyVariable* may be contained in the *qosPolicySimpleCondition* instance
  - the class *qosPolicyVariable* may be referenced from the *qosPolicySimpleCondition* class
- Value Representation
  - the *qosPolicyValue* class may be contained in the *qosPolicySimpleCondition* class
  - the *qosPolicyValue* class may be referenced from the *qosPolicySimpleCondition* instance

QoS Policy Variables

QoS policy variables specify the attributes of a flow that should be matched when evaluating the condition. Variables also constrain the set of values within a certain value type that can be matched against it in a condition. For example, a variable for the source port number must constrain the set of values to the valid range of port values integers. QoS policy variables are also are used for binding individual conditions. The three main attributes for a value are:

- *qpVariableName* of the *qoSsPOlicyVariable* class which specifies the well known name used for logical binding
- *qpValueTypes* represents the list of value classes that can be matched to a particular variable
- *qpValueConstraints* representing a list of constraints on a particular variable

Meta-data, obtained by combining the above main attributes, define the semantics of variables to be used to form QoS conditions.

In order for different policy management systems to interoperate the same variables must be initialized to the same value by different policy servers. Even if their binding mechanisms can be different, the binding logic must give identical instantiation. Each variable must be bound to a logical entity. When a policy server evaluates an expression containing variables, it must first instantiate them. To be instantiated, a variable must be bound to a specific value and associated with a logical entity. Some variables are pre-defined.

2.5 Implementation Issues

After having seen the current IETF policy framework, we are now considering the main implementation issues related to policy-enabled network management.

2.5.1 Policy-enabled tools

Management tools have to provide ways to facilitate the specification of roles, the defining of conditions, the detection of conflicts and so on. Inside a domain, policy conflicts can be managed in an easier fashion, subject of ownership and focused goals. In the case of interdomain policy conflicts, one should possess some tools capable of detecting and resolving at runtime potential conflicts raised by different ownerships and, eventually, different goals.

2.5.2 Policy-oriented languages

The system management seeks the provision of the highest QoS possible to the users. The two steps necessary to achieve this goal are: define the required QoS in terms of measurable properties and define suitable policies to ensure the desired QoS. Currently, most management activity is done by the network managers who monitor the system's status, analyze the situation and take the appropriate actions to correct errors or improve the system's functioning. Several problems arise when management occurs in this manner [Koc96]:

- different managers may interpret differently similar situations resulting in different actions
- management knowledge is not documented , so each manager has to find his/her own solution
- network managers put a lot of time in reactive action to urgent problems.

To improve this situation, automation of the network management task is required.

For the basic formalization of various attributes and MOs, ASN.1 and GDMO can be used. However, an appropriate policy oriented language must allow a user designer to express new requirements such as events, relationships between actions, roles, and temporal issues.

Management policies are specific to each functional area of network or system management [Wei95]. Alpers and Plansky presented the domain-based management policies and a GDMO-based formalism within a policy-platform [Alp95].

Many approaches consider the policy as class schemas. For example, in Object – Z, one can specify the managed objects. A similar approach allows us to define an object representing a policy. A specialization of this schema leads to different policies. A policy has its own type. The most generic policy type inherits from the software type and has specific static features, such as the domain belonging to, the time interval it holds, its target referring to the concerned managed object, the scope defining the level of validity starting from the target base within the Object Oriented hierarchy, and the policies which are concurrently used. Other policies could inherit by specialization from this generic type. As dynamic properties, a policy has the refinement status, trigger mode, and administrative state. The refinement represents the degree of maturity of a policy, which can be definitive or experimental. The trigger mode reflects the procedure of applying a policy. The administrative state is usually unlocked. Locked state is used when testing a policy or when modifying it by meta-policies. All these properties are visible and can be modified according to other policies. The important difference between the behavior attribute in GDMO templates for managed objects and a policy behavior is that the managed object behavior defines possible or available behavior of the resources they represent expressed by their state values, while a policy behavior is a restriction on the possible behavior expressed as the desired behavior. The desired behavior of a managed object is dependent on different goals, whereas the real behavior of a managed object depends on the behavior of its cooperation relations and managed objects cooperating with it.

The policy as an object approach is taken by standards [Iso19]. They focus on policies for a management domain. A policy is represented by a managed object having the following features: name, generic contents, and a set of domains to which the policy refers to. From the management point of view, a policy encapsulates a representation of system management goals. Other commercial products present some « rudimentary functionality to define domains and apply policies » [Wei95]. For example, the HP's Dolphin uses an object-oriented Prolog-like language to specify policies [Pel93]. The TME (Tivoli's Management Environment) defines policy objects as a set of customizable programs written as shell or Perl scripts [Wel94].

A rule-based policy is presented in [Iso19]. A policy is a set of rules of the form :

$P1 \in \delta^1$

$P2 \in \sum(P1) + \delta^2$

$P3 \in \sum(P1) + \delta^3$

$P4 \in \sum(P2,P1) + \delta^4$

where P4 is the most specialized version of a policy P1 and δ^n represents discrete rules specific to that policy. In P4, additional rules to those specified by P1 and P2 cannot contradict any rule specified in either P1 or P2. Here, a correlation between δ^4, P1's rules and P2's rules must ensure that there are no conflicts. Koch and al. present a similar approach [Koc95].

2.5.3 Interoperability between policy-enabled management tools

We have seen that, in fact, policy-enabled approach may refer to distinct technology networks (ATM. IP, Frame Relay). Management tools must cover all these.

Many specification languages can be used and, eventually, many different CIMs or the same CIM being partially and, perhaps, differently implemented. Therefore, there is an increased need for tools coping with these issues.

The next chapter focuses on policy-enabled tools and the solutions they offer to solve policy-enabled network management problems.

Chapter 3

Policy–enabled Tools

3.1 Management Tools

From the large set of management tools (Figure 3.1), one can distinguish a subset of policy-enabled management tools. Their particularity is that they introduce means for defining, editing, deploying and reusing policies as a way to manage the network. A subset of policy-enabled management tools is formed by target-oriented policy-enabled tools; these tools are designed specifically to suit particular applications, for example, video-on-demand.

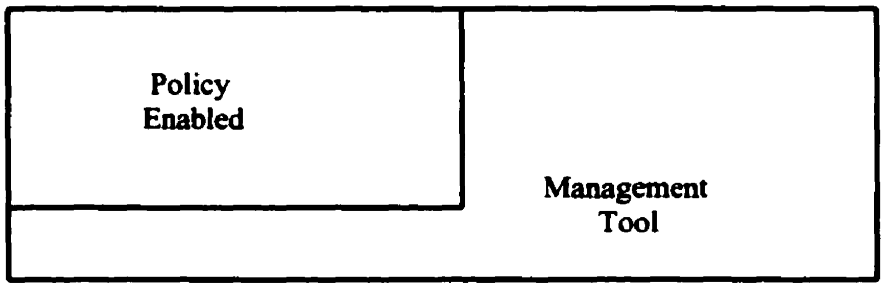


Figure 3.1 Management tools and policy enabled tools

The transition from classical tools to policy-enabled tools requires more than a simple standard body effort. We estimate that we are facing a new management paradigm. Implementing it signifies a new information model and decision model, and needs appropriate protocols and new hosting and access mechanisms. The current tendency is to open network elements via standard APIs (Application Programming Interface), and to allow the push model to appropriate policies across these APIs.

Before analyzing the major players in the policy-enabled-management, we will first consider the relevant features of a policy-enabled tool. Following is a "wish-list" of the desired characteristics of a policy-enabled-tool.

1) *Upgradability* – The tool should allow new mechanisms to be incorporated in its list of features.

2) *Standard / IETF drafts – orientation* – The tool should operate according to pre-defined and widely used standards, such as the Common Information Model (CIM).
3) *Scalability* – The software should be scalable in terms of:
   (a) the number of policies (should allow the specification of the number of policies);
   (b) the types of policies (should account for different policy types and allow the definition of such various types), and
   (c) the number of network elements covered (should allow the specification of the number of network elements to be affected by a particular policy).
4) *Utilization of Relational MIBs or PIBs.* The classical version of network management tools use MIBs, but policy-oriented software should use PIBs as their information bases.
5) *Legacy Protocols or COPS* – Policy-enabled tools should use policy-oriented communication protocols such as COPS rather than legacy protocols such as SNMP.
6) *Policy definition GUI* – The software should provide a graphical user interface to allow the manager to define policies.
7) *Policy Conflict Detection* – Policy conflict detection involves the verification of rule consistency, and the detection of both condition and action conflicts. Some tools do offer such features [Lup97] whereas others don't.
8) *Temporal properties concerning policies* – A policy-enabled tool should allow the expression of time or of the relative position of two events, including the time intervals.
9) *Correlation between temporal properties* – Within a given time zone, correlation adjustment between temporal properties is not necessary. However, between time zones, mechanisms that handle correlation between temporal properties should exist.
10) *Error-Handling Mechanisms* – Appropriate error-handling mechanisms should exist to deal with situations in which policies fail partially or entirely.
11) *Network Dependence / Independence* – Some policy-enabled tools are dependent upon the presence of particular elements in the network and on the network topology, whereas others are independent of the network they function on.

12) *Interoperability with other existing tools* – When two different network interoperate, the software tools on each of them should also interoperate.
13) *Proactive mechanisms* – Policy-enabled tools should have measurement features and policies to prevent certain event types such as network failure, CPU shortage, link bandwidth saturation, etc.

3.2 Generic Management Tools

In order to satisfy the previous list of features, classical generic management tools were developed. They are robust in representing the network topology and have a friendly operator GUI, although they offer very few automated mechanisms.

3.2.1 Intel

Intel's network management solution, named Intel Policy-Based Network Management, uses policies in order to implement Service Level Agreements (SLAs) by establishing contracts with user groups.

Using a single console with a simple GUI, managers can define, edit and deploy policies for multi-vendor networks. Another advantage is the use of COPS to support heterogeneous network environments and legacy network elements. The software is also scalable to accommodate future changes in network resources [Int00].

With Intel Policy-Based Network Management, policies are stored in directories, which are not policy brokers, but only common repository for user information. COPS, LDAP and RSVP standards are supported.

3.2.2 Tivoli

Tivoli's management solution, Tivoli Remote Control, regards policies as a set of rules to be applied to managed resources in order to control the default values of newly created resources (default policy) and to maintain the guidelines when administrators modify or operate on resources (validation policy). An example of Tivoli policy is a rule requiring user login names to be eight characters or less.

Tivoli uses policy regions as containers for managed resources that use the same set of policies. These regions also serve as basis for assigning administrator permissions or roles. Policy regions enable Tivoli Remote Control to provide granular access control, which is necessary to implement remote control in large organizations. The ability to centrally manage remote control policy makes the product scalable - the complexity of management does not increase exponentially with an increase of the size of the network.

Tivoli Remote Control's GUI enables quick access to the available commands and the display of the current state of the session.

Other features supported by the software include: cross-platform support for controller and target: Windows family and OS/2, gateway support to provide efficient and secure access through firewalls or across different networks, DHCP support and dynamic target lists. On the other hand, in terms of policies support, the software does not provide error-handling or proactive mechanisms, temporal properties or correlation between temporal properties or other policy conflict detection features [Tiv97].

3.2.3 Sun

Finally, Sun's Bandwidth Allocator 1.0 allows for the easy specification of bandwidth allocation policy. At the same time, it provides reporting utilities to follow how bandwidth is used and how efficiently the traffic provisioning rules are working.

The *Packet Classifier* component of Sun's Bandwidth Allocator takes an outgoing packet and assigns it to a traffic class based on the information found in the IP or application header. Some of the factors that are important when assigning a packet to a particular class are: protocol type, IP source and destination address, TCP/UDP source and destination port. The assignment of quality of service is done according to a provisioning rules file, which can be updated by command-line mode or via a Graphical User Interface (GUI). The *Packet Scheduler* component transmits different classes of traffic according to the pre-defined priority and transfer rate for each class. Statistics collected during the classification process are available via a command-line interface or via GUI through a proprietary MIB. Monitoring can be done via SNMP.

The software uses the following filters for the provisioning rules:

(1) traffic type (TCP/IP or UDP ports or services)

(2) source or destination IP address (allowed discrimination between different machines or organizations)

Filters can be placed in a hierarchy. For example, a pre-defined quality of service can be assigned to traffic coming from a particular organization. Within this traffic, a subset of the pre-defined quality of service can be reserved for FTP, for example [Sun98].

3.3 Special Policy-Enabled Tools

As outlined in the introduction of Chapter 3, a particular subset of the management tools available, are policy-enabled tools. Policy enabled network management relies on many protocols. Each vendor of policy-enabled management tools must decide how to implement the communication between many policy servers, directories and the entities being managed. Following, we present a non-exhaustive list of special existing policy-enabled tools with respect to the degree of satisfaction of the previously outlined features wish list.

Some policy-based tools are vendor oriented because vendors are building software tools specific to their own equipment. In this category we consider Cisco's, Lucent's, Hewlett Packard's and Nortel's, whereas others are more generic in both types of networks and problems to be solved (Allot, Orchestream, Extreme Networks, IPHighway, Spectrum).

3.3.1 Cisco Systems

Cisco Systems QoS Policy Manager 1.1. (QPM) and User Registration Tool (URT) 1.2 (beta version) are two packages of software designed to solve policy-enabled networks management problems.

QPM is a configuration interface for Cisco devices. It enables differentiated services for Web-based applications, VoiceOverIP, Internet appliances, and business-critical processes, ensuring the desired QoS. It automates QoS configuration and deployment and improves multi-service performance. QPM allows the definition of

QoS policies at a more abstract level than can be defined using device commands. For example, one can define policies for groups of devices rather than one device at a time. Policies can be created that apply to applications or groups of hosts.

The QPM *Policy Manager* allows the definition of QoS policies for devices or interfaces, and sets the queuing mechanism for interfaces. Policies can be created for groups of devices, interfaces, hosts, or definitions of application traffic. Web-base reporting to helps the maintenance of policy definitions. The *Distribution Manager* allows the deployment of policies and QoS configurations created by the Policy Manager to the network devices. Loggings and web-based reportings help to maintain records of policy deployments. The *QoS Manager* – distributes the changes requested by the Distribution Manager, configures devices and maintains the QoS database created by the Policy Manager [Cis00].

Cisco's QPS strength relies in its mechanism to define multiple conditions. This condition matrix supports: source and destination IP address, IP subnet mask, protocol type (IP, TCP, UDP), application port numbers, and IP precedence.

When defining a policy, a manager selects an interface or group of interfaces from the management console and defines the type of queuing mechanism to be used. Roles are handled on a per-interface basis. The interfaces support a variety of roles including priority queuing, custom queuing, weighted fair queuing, weighted random early detection and class-based weighted fair queuing. The actions include coloring a flow with IP precedence information, limiting the bandwidth to a particular value, apply proritization by directing the traffic into a particular queue and custom queuing. However, issues such as temporal policy conflict detection are not addressed.

Cisco's URT software allows user-tracking mechanisms in Windows NT and NetWare NDS-enabled IP environments. It uses a combination of server and client technologies to dynamically track a user at login, assign that user to a virtual local area network and track the user's location.

3.3.2 Lucent Technologies

Lucent Technologies proposes RealNet Rules (Beta Version), a new policy management application for multivendor environments that allows managers to drive network behavior that maps to business needs. Unlike Cisco's solution, Lucent builds its policy based management software around the LDAP interface. The software focuses on three main points of policy-based control: QoS, security, and resource allocation. These provide an automatic mapping of users to available local resources. It offers the inherent scalability and fault tolerance that solutions using COPS or CLI have to build in manually.

RealNet Rules reduces complexity by providing an intuitive graphical user interface. Network managers are allowed to implement high-level network rules into detailed configurations, which are then applied to the network. Policies are stored in a directory and can be acceded via LDAP-enabled applications [Luc99].

RealNet Rules is directed at the enterprise LAN. The software uses the CIM directory schema to store policy in the LDAP server and allows applications to manipulate and use this policy data. RealNet Rules supports conditions such as IP source and destination addresses, layer 3 protocols, layer 4 port numbers, Diffserv code points. However, the solution addresses only the basic features of end-to-end policy management, excluding features such as dynamic applications, different types of queuing mechanisms, provisioning protocols (HTTP/CLI/COPS), DiffServ/ToS markings or temporal policy conflict detections [Con99].

3.3.3. Hewlett-Packard

Hewlett Packard's HP OpenView PolicyXpert allows network managers to automate the configuration of QoS mechanisms across a heterogeneous network environment by capturing them in a high-level terms. A single set of policies can be deployed across many managed elements, so that there is no need for configuring QoS on each element individually. The software supports three QoS abstractions that can be implemented using various QoS mechanisms. Prioritized Class of Service policies can use queuing, IP Precedence marking, or IEEE 802.1P frame tags to define a class of service for application traffic. Assured Bandwidth policies can use TCP rate control, class-based queuing, and committed information rate mechanisms to assure bandwidth. RSVP policies can include allow or disallow, maximum bandwidth, and flow priority to control applications that signal QoS requirements using RSVP [Hew00].

PolicyXpert's conditions can be complex combinations of one or more of the following parameters: time, date, day of the week, application port number, source or destination IP address, IP type of service, HTTP URL or VLAN ID. Traffic can be allowed or denied based on the RSVP admission control model. When a condition is matched, a usage model can be applied to that flow. The main advantages of PolicyXpert are:

- its multivendor devices support ;
- its RSVP capabilities, and
- its use of COPS to provision network devices.

When COPS is not found on a target platform, a proxy agent acts to configure the non-COPS-compliant device [Con99].

Some of the software's disadvantages include:

- it does not provide any filtering or security features ;
- it has no LDAP integration, unlike Lucent's solution, and
- does not provide any temporal policy conflict detection mechanisms.

3.3.4 Nortel

Nortel's network management solution, Optivity Policy Services 1.0, is built around an Oracle database and, like Lucent's solution, works together with both policy management and IP address management.

When dealing with policy management, Nortel groups devices into various domains. The software uses a Java interface to access Nortel BayRS and Cisco IOS router platforms, but one interface can be member of only one domain, which limits the types of policies that can be applied to the network. Like many of the solutions offered by other vendors, Nortel's solution has a condition/action model, including all main IP layer 3 and 4 conditions.

The software has the ability to deny service, mark flows with a particular class of service coloring, regulate and monitor traffic flows and shape traffic to a specific bandwidth. However, Nortel's solution doesn't allow queue behavior configuration. SNMP and CLI are presently the communication means, but their replacement with COPS is part of future improvement plans [Con99].

3.3.5 Allot Communications

NetPolicy, the policy-enabled tool of Allot Communications, is a Java-based application that enables the deployment of complex policies in already existing networks [All00]. The software uses a unique, table-like interface to define conditions and actions needed for policy definition, but does not treat roles separately as other available products do. NetPolicy can define flows in terms of the IP addresses and netmasks, groups of IP addresses, types of protocols, TCP/UDP port numbers, IP ToS information, time of the day and other high-level information. Examples of actions the software supports are: allocation of minimum and/or maximum bandwidth, guaranteed bandwidth, priority and limitation of number of concurrent sessions.

NetPolicy works very well in combination with other Allot network hardware. It supports load-balancing and cache-redirection based on policy. NetPolicy's modular architecture works together with Allot's policy enforcement devices (NetEnforcer) to ensure service and user provisioning, policy enforcement, service verification and billing.

The software does support multiple platforms, but the features supported for each platform are restricted to queuing and access control list. It allows for the management of different hierarchical groups. Single flows can be manipulated independently and groups of flows can be acted upon as a single logical unit by defining group policies that apply to a collection of flows.

Another feature present in NetPolicy is an active feedback mechanism by which network administrators are able to see in real time the effects of applying their policy on the network. COPS and CLI are used for communication [Con99].

NetPolicy creates a complete policy system intended for both Internet Service Providers and for enterprises. For the former, it gives the ability to provision, enforce, verify and bill service level agreements and to offer bandwidth on demand and bandwidth brokering services. For the later, NetPolicy gives a unified view of QoS policies and network actions in real time.

3.3.6 Orchestream

Orchestream Enterprise 2.0, Orchestream's policy-management solution, offers a complete set of conditions, actions and roles. It is the first software capable of automating the deployment of QoS and security features across the enterprise.

Orchestream is the only product on the market based on the new IETF DiffServ framework [Orc99]. One can assign to each device or interface a DiffServ role when they are placed on a network domain, without the need to upgrade the network hardware or software.

Each application flow is allocated the appropriate level of bandwidth by aggregating flows into a small number of classes. The network manager has complete control over the allocation process.

The Orchestream software is designed to support multi-vendor environments such as those from Cisco Systems, Xedia, and Lucent. Orchestream Enterprise 2.0 provides policy-based control of access security, allowing all existing routers to be configured into packet-filtering firewalls in one step. Devices can be configured by SNMP, HTTP and TACACS+ (Terminal Access Controller Access Control System Plus). At the same time, its QoS policies are equipped with protection against denial of service attacks [Orc99].

3.3.7 Extreme Networks

ExtremeWare Enterprise Manager 2.0 (Beta version) of Extreme Networks is a complete management tool enables you to set policies that provision bandwidth and prioritize applications for specific end-user groups. It allows network-wide inventory control, management and configuration of multiple Summit switches, the Summit Virtual Chassis, VLANs and Policy-Based Quality of Service (QoS).

ExtremeWare Enterprise Manager uses a three-tiered architecture. The platform is very extensible and allows the use of SNMP, Java, HTTP and standard SQL relational database management system. It also supports ExtremeWare's SmartTraps™ feature, which minimizes bandwidth usage by enabling Extreme switches to interact with the ExtremeWare Enterprise Manager platform [Pro98].

The software defines conditions based on physical port, a specific VLAN, a specific IP address or subnet, a particular Layer 3 protocol and Layer 4 TCP or UDP port or group of ports. Using DLCS (Dynamic Link Configuration System), the software is able to import users, groups and workstations in a Windows NT Primary Domain Controller (PDC) environment. When loggings are executed in or out the PDC, the user is tracked and reported to the policy server.

The software lacks some granularity because it applies the policy to an entire device rather than on the per-interface basis. It also supports third-party network management system software integration via HP OpenView [Con99].

3.3.8 IPHighway

Unlike Orchestream, IPHighway is taking a more strategic approach to policy-based networking by giving internally developed source code for COPS to any vendor willing to implement the protocol on its devices. COPS is implemented to manage every device that supports the protocol.

IPHighway's network management solution, Open Policy System 1.0, has a friendly policy definition GUI with concise definitions of policies and actions. Among the conditions for policy management we cite: source and destination IP address, application type and physical interface. A feature specific to IPHighway's Open Policy System 1.0 is the enabling of several unique groupings of subnets and interfaces such as « all interfaces supporting weighted-based queuing ». The actions that the software supports include: FIFO queuing, WFQ (Weighted Fair Queue), custom queuing, priority queuing, class-based distributed fair queuing and an IPHighway hybrid queuing mechanism that uses both WRED (Weighted Random Early Detection) and custom queuing.

On the negative side, the product does not support COPS functionalities, nor does it deal with policy conflict detection or multi-vendor support [Con99].

3.3.9 Spectrum Management

Finally, Spectrum Management's solution, the Spectrum Policy Based Network Management Suite, takes advantage of the Cableton's SecureFast technology base. The switch reports information such as IP and MAC addresses that are attached to a particular switch port in order to simplify policy management.

The advantages this software offers include: a large range of conditions (ethertype, source and destination port numbers, IP ToS information, IP protocol type, IP source and destination addresses, IPX (Internetwork Package Exchange) class of service, IPX packet type, IPX port number), a large range of actions (traffic assignment to a particular VLAN, prioritizing packets into multiple queues, coloring packets with IP ToS information, packet filtering and rate limiting). The software uses SNMP and CLI as communication means between the manager and the managed objects. Also, LDAP and LDIF (LDAP Directory Interchange Format) inform the manager of the changes to the directory.

On the drawbacks side, we can mention the lack of multi-device management, the lack of policy conflict detection support, the lack of temporal properties concerning policies, and its dependence on SNMP and CLI to perform the job,.

Considering policy-enabled tools, a few features concerning policies have been partially implemented or are part of future implementation plans. Table 3.1 summarizes some of these particular characteristics.

Name	LAYER					Temporal Issues	Policy Actions		
	1	2	3	4	5 +		Various Types of Actions	Types Of Queuing	Protocols
Allot	Yes	No	Yes	Yes	Partially	Yes	Partially	Partially	Yes
Cisco	No	No	Yes	Yes	Partially	No	Partially	Yes	Partially
Extreme Networks	Yes	Partially	Partially	Yes	No	No	Partially	Partially	Partially
Hewlett Packard	No	Partially	Yes	Yes	Partially	Yes	Partially	Partially Future	Partially
IP-Highway	Yes	No	Yes	Yes	No	Yes	Partially	Partially	Partially Future
Lucent	No	No	Partially	Yes	No	Partially	Partially	Future	Partially Future
Nortel	Future	Yes	Yes	Yes	Yes	Yes	Yes Future	Yes Future	Partially future
Orche-Stream	Yes	Partially	Yes	Yes	Future	Yes	Partially	Yes	Partially Future
Spectrum Manag.	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes

Table 3.1 Summary of policy-enabled management tools features

3.4 Conclusions on Existing Tools

We conclude that there is a paradigm shift in network the management community, the industry and the service providers. The current achievements allow us to predict more emphasis on inter-domain correlation, adoption of COPS-like protocols for policy manipulation, etc.

A good policy management strategy can be divided in four steps.

1) Identification of the type of network traffic using classical tools such as RMON, SNMP or other packet-capturing tools. This includes measuring the amount of bandwidth used, the peak load times, traffic burst sizes, and packet size distribution. The latency applications tolerate is also important. None of the products we presented offer these functionalities yet.
2) Construction and deployment of a set of policies to help bandwidth and different types of traffic management. Most of the solutions we examined allowed the dynamic creation of policies and provided means to shape the traffic to meet various needs. The most dynamic solutions were offered by Cisco and Orchestream. Also, Hewlett Packard and IPHighway allowed multi-vendor policies.
3) Deployment of mechanisms to measure the effects of defined policies. Feedback mechanisms can constitute a first solution, but, ideally, network elements should have the capability to report service level statistics to the policy-enabled tool. Allot's was the only product solution from those considered that provided feedback on the deployment of defined policies onto the network.
4) Network self-healing and self-tuning. None of the solutions presented offered such a feature. A first step towards these characteristics would be the reporting based on device statistics. Spectrum Management did offer a reporting mechanism, but not the necessary statistics to determine if the policies were really working.

We have identified that some improvements still need to be made. Summarizing, there is a special need for correlation algorithms to deal with conflicts arising from various situations such as different policies referring to the same object and triggering conflicting actions, different policy owners, having opposite interests, setting different pre-conditions for the same set of actions, the lack of companion policies, the same policy triggering conflicting actions: parallel actions, sequential actions or overlapping actions, or different policies triggering conflicting actions.

Sometimes a policy is triggered by an event or set of events that already caused a network failure. In this case, it is too late for a policy-based remedy. Prediction mechanisms in order to forecast the change tendency of the network state must be used in such situations. Usually, this is done by a state refinement (warning thresholds) preventing a policy system to be late. For example, a refinement of the operational state *enabled* can be specified by another state attribute called *UsageState (idle, active, busy).* By interpreting these values, the managing system may either apply policies in the subsystem with respect to the state of one particular component or choose to react directly across an administrative state attribute of this component, by the use of commands. As opposed to the *operationalState (enabled, disabled)* attributes whose values are updated by the resource itself, commands act on attributes whose values are updated by the managing system [Bir96].

We estimate that temporal issues have only partially and insufficiently been considered in the tools we have studied. We will consider further the particular temporal issues related to:

(a) Particular requirement from a policy to be fired only under the presence of another policy declared as valid and fireable or already fired.

(b) Dealing with Time Zones. When a policy and its required policies are in the same time zone there is no temporal translation needed. When the required policies used by another policy are defined in a different time zone, the necessary time zone adjustments must be done. When a policy is defined in a particular time zone to be fired in another time zone, again, the time zone appropriate modifications must be effectuated.

(c) Dealing with temporal relative dependency between actions.

    (i) the question is how to specify this kind of dependencies in order to capture actions ordering, actions overlapping, or other relationships between the start/duration/stop temporal expressions.

    (ii) This kind of conditions must be checkable with respect to the effectiveness of their scope. Some systems have sensors for capturing the fact that an action succeeded. Usually, management tools require another "check-status" action on the MIB to which the target object belong, via *get/get bulk* (SNMP) or *m-get* (CMIP) etc. Practically, when a policy partially fails on a non-synchronism between its actions, the entire subnetwork must be set.

We will consider these aspects to improve the existing policy CIM.

Chapter 4

Proposal for Temporal Policy Correlation

Policies need a richer CIM, especially in order to express types of events, types of interactions between networks as well as types of policies that can be applied upon a network.

Different correlation mechanisms and specification techniques can be used to detect conflicts when policies are applied (temporal logic, temporal interval logic, correlation windows).

Due to the real time issues of different events that occur in networks and to the fact that networks are widely spread on the globe, it is important to synchronize the actions that result from policies with apparently non-related goals when applied.

4.1 Typing Schema

Generally, events occurring within a network are well typed (SNMP traps, CMIP notifications, types of alerts etc). Most of the component parameter values are dependent upon various kinds of relationships a given network component has with its neighbors (since interactions take effect across these relationships). It is logical that for a given combination of different types of events occurring within a system, and considering various component types and component interaction types, different policy types apply.

4.1.1 Typed Events

According to a given activation event type, a policy may trigger a given action or set of actions. Basically, we will present the (i) SNMP (RMON) *alarmGroup* with different types of events and a useful hysteresis mechanism for triggering, (ii) a solution for fault (alarm) notification that prescribes what a normalized set of alarms must include, and (iii) summarize how CORBA Event Service treats the event typing issue.

Remote Network Monitoring (RMON) defines standard network-monitoring functions and interfaces to communicate between SNMP- management consoles and remote monitors. In the context of RMON, the *alarmGroup* is used to define a set of thresholds for network performance. When a threshold is crossed, an alarm is sent to the monitoring console. The *alarmGroup* consists of an *alarmTable* whose entries are variables to be monitored, sampling intervals and threshold parameters. The objects in the *alarmTable* are:

- *alarmIndex* – unique integer representing the row number in *alarmTable*
- *alarmInterval* – interval in seconds over which data are sampled and compared with the thresholds
- *alarmVariable* – object identifier of a particular variable in the RMON MIB to be sampled. The only object types allowed are those that resolve with to ASN.1 type *integer*, namely *integer*, *counter*, *gauge*, and *TimeTicks*.
- *alarmSampleType* – the method of calculating the value to be compared with the thresholds. When the value of the object is *absoluteValue()*, then the value of the selected variable is compared directly with the thresholds. When the value of the object is *deltaValue()*, then the difference between the value of the selected variable at the last sample and its current value is compared to the thresholds.
- *alarmValue* – the value of the statistic at the last sampling period
- *alarmStartupAlarm* – the possible values are: *risingAlarm()*, *fallingAlarm()* and *risingOrFallingAlarm*. Its value determines if an alarm will be generated if the first sample after the row becomes valid is greater than or equal to the *risingThreshold()*, less than or equal to the *fallingThreshold()* or both.
- *alarmRisingThreshold* – the rising threshold for the sampled statistic
- *alarmFallingThreshold* – the falling threshold for the sampled statistic
- *alarmRisingEventIndex* – index of the eventEntry used when rising threshold is crossed.EventEntry is part of the eventTable of RMON.
- *alarmFallingEventIndex* -- index of the eventEntry used when falling threshold is crossed.

The event types in *alarmSampleType* are usually established by considering their occurrence (*periodical, non-periodical*). For example, according to their incidence in the system, alarms can be categorized as yellow alarms, specifying warnings or red alarms, specifying outstanding alarms.

Small fluctuations in the values are prevented from causing alarms by a process that RMON calls Hysteresis Mechanism (Figure 4.1).The alarm generation mechanism is viewed as a two-states process. In the rising-alarm state, a rising alarm will be generated when the value of the observed variable reaches or is higher than the rising threshold. In this situation, the mechanism is disabled from generating a falling alarm. After the rising alarm is generated, the mechanism enters the falling-alarm state and remains there until the observable value reaches the falling threshold [Sta93].

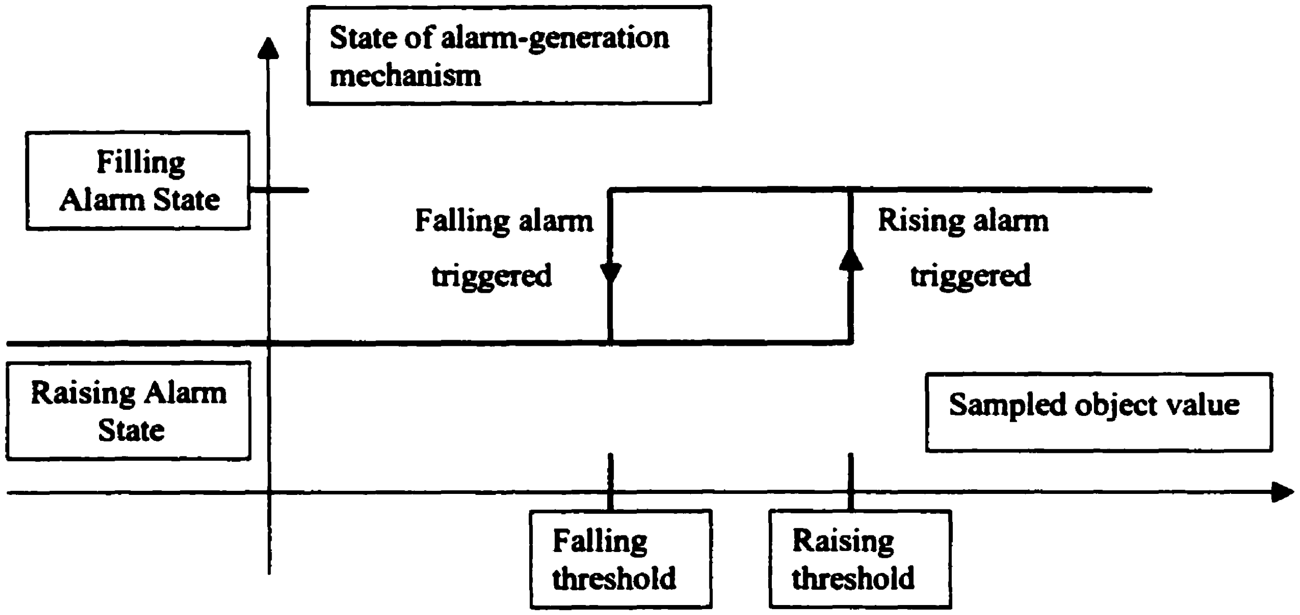


Figure 4.1 Hysteresis Mechanism

Houck and Finkel [Hou95] present an alarm notification system, commercially available under the name of NetFACT software. The idea is to use a collection of techniques specialized to the type of topology that encompasses the faults or alarms being considered. The two techniques used are:

- *path analysis* – deals with failure of path problems. Faults are reported by alarms, which are then analyzed to evaluate which component of the path is faulty, and

- *tree search* – used to determine if the nodes identified in path analysis are directly responsible of a fault or if they are part of a faulty component.

The different stages of problem solving with NetFACT are briefly presented below:

- Awareness – build the internal representation of the alarm and wait for related alarms to arrive;
- Get Config – obtain the configuration information from the configuration model;
- Diagnosis – identify the cause of the problem by the techniques presented above;
- Recovery – await the recovery of network components affected by the fault;
- Closure – mark the problem as closed, and
- Purge – an operator purges the problem from the system.

Upon the receipt of an alarm, its corresponding model object is located in the database. A new alarm must be normalized to a consistent syntax and semantics. A normalized alarm must include:

- the identity of the object to which the alarm refers;
- the impact to the behavior of the object;
- votes representing the probable source of fault, and
- other information such as alarm ID and timestamp.

Another event model [Sch97] is used in CORBA-based monitoring and management systems. In an object-oriented framework, event classes specify the attributes contained in event messages of the corresponding event class type. The definition of the class *Event* contains the generic attributes of notification messages common to all instances of event classes, which can be derived from this abstract base class. Two kinds of attributes may be assigned to each event: *dependent* or *independent*. The dependent attributes are determined by the particular type of event. The class *Event* represents the root of inheritance tree formed by subtyping to describe events emitted my real MOs. For this reason, event channels can be used as *typed event channels* based on the CORBA Event Service [Omg95] using Event as the expected type for push server and consumer interfaces.

4.1.2 Typed interactions between network elements

Relationships are class properties that represent the dependencies between MOs. Therefore, all objects instantiated from a given class have similar relationship types. Fault propagation in the network is dependent upon the network elements relationship dependencies. Next, we will present a three levels dependency model:

- the first level is the direct in indirect dependency class;
- the second level is composed of three relationships which inherit from the first level : *users and shares*, *connection end-point* and *matched connection*, and
- the third level is composed of five dependency relationships : *usage allocation dependency*, *message passing dependency*, *shared resources dependency*, *existential dependency* and *operational dependency*.

First Level:

MOs in telecommunication networks are dependent upon each other in complex ways. All dependencies are composed of two kinds of basic dependency (Figure 4.2)

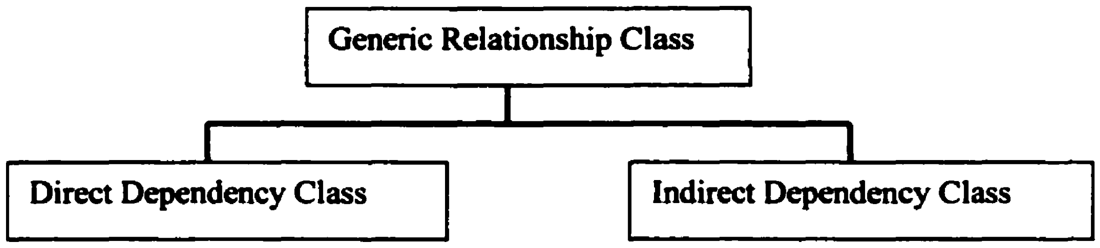


Figure 4.2 Direct and Indirect dependency Class

In the *Direct Dependency Class* an object is referenced directly by another object and appear as directed edges in the dependency graph (Figure 4.3). In the *Indirect Dependency Class*, a sequence of one or more direct dependencies exists between two objects. In a dependencies graph, if there is a directed path from A to B, then A is said to depend on B and B is said to be a supporter of A. If a node is dependent upon itself, then there is circularity. A node A is a candidate supporter of B if the addition of a direct dependency of B on A does not induce circularity.

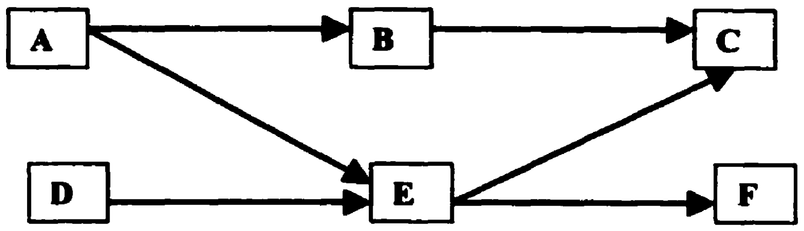


Figure 4.3 Example of Dependency Graph

In the Example of Dependency Graph (Figure 4.3):

- A depends directly on B and E
- A depends on B, C, E, and F
- E is a supporter of A and D
- D is a candidate supporter of A and B, but not of C, E or F

Second Level:

Three-second layer dependency relationships have been identified [Jor93]. They inherit the features from the superclass direct or indirect dependencies. This means that network elements that have these three relationships can depend directly or indirectly on each other. Indirect dependency is the result of direct relation composition. Figure 4.4 depicts a n-layer connection between objects A and B with corresponding underlying connections and links used to realize this connection.

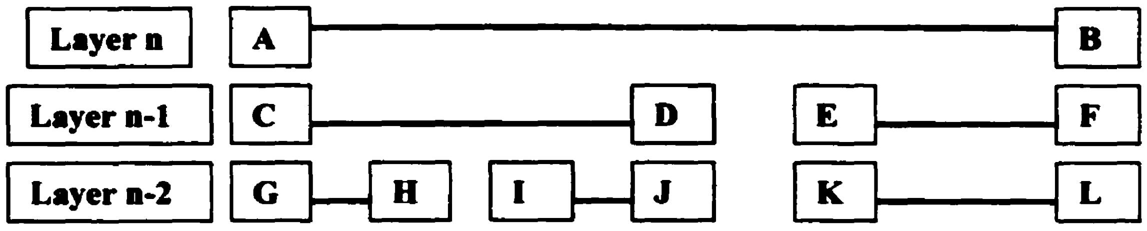


Figure 4.4 n-Layer Connection

(a) *Uses and Shares Relationship* relates the resources to the hardware or software resources on which they are based. For a n-layer connection, the application of this relationship will give the local view of the (n-1)-layer connection, which is used for transmission. In the example, object (A, B*) uses* (C, D) and (E, F). The layers in the n-layer connection example do not reflect the OSI communication protocol layer, but any organization in which the objects have "*uses*" relationships between them. When two objects use the same resource, they have a "*shares*" relationship between them based on "uses" relationship.

(b) *Connection EndPoint Relationships.* Two MOs representing endpoints are connected directly and the superclass is *direct dependency class.*

(c) *Matched Connection Relationship.* This relationship exists between peer connections, which are linked by gateways or switches. The superclass is *direct dependency class.*

Third Level:

Level 3 contains terminal dependency classes in the dependency hierarchy we are presenting. They are subclasses of classes from the second level which inherit from the direct and indirect dependencies. Level 3 contains the following dependency models : *usage allocation dependency, message passing dependency, shared resources dependency, existential dependency* and *operational dependency.*

Very few existing management tools consider all these three dependency models. If we reconsider the NetFACT (Section 4.1.1), fault visibility can be limited and not all the dependencies that point to a cause of fault are known. Only applying the NetFACT heuristic may result in incorrect identification of a node that is visible for NetFACT as a fault cause, while the real fault cause is a hidden common element inside the carrier network. To solve the problem, such failures are reported as independent faults. Hidden dependencies called for a change in NetFACT's heuristic from "minimum number of faults" to "best explanation for each symptom".

Another kind of interaction between network elements is also found in the NetFACT example and concerns the nature of dependency relationships between network elements. In the case of simple dependencies, the dependent resource depends only on the binary availability of the supporting resources. More complex dependency relationships depend on logical combinations of other resources. Other resources depend on a quantitative amount of capacity in the shared resource. In such cases, a drop in the available capacity may cause failures in the dependent resources.

4.1.3 Typed Policies

As already outlined in Section 2.1, policies range from high level, that is, abstract, non-technical policies, to low level, that is, technical policies, immediately instantiable and applicable. Usually, these low level policies specify as resulting actions some actions from those offered by SNMP, COPS and LDAP, already presented. The level of abstraction is a function of the degree of detail present in the policy definition and the ratio of business issues to technological issues within the policy.

Policy classification has a series of advantages, such as:

- better understanding of what exactly can be accomplished through management policies;
- identification of similarities and differences between policies to determine their corresponding class;
- availability of a policy hierarchy for policy transformation process, and
- derivation and verify the components of formal definition of policies.

Since we promote a system with typed events and typed dependencies, a typed policy-enabled tool will ease policy conflict detection and resolution. This also depends on the manner policies are specified and stored. Some service providers such as FidoNet or VirNet maintain their policies in informally written policy catalogues. Rene Wies [Wei95] presents multi-dimensional criteria for policy classifications, the result of deep analysis of these policy catalogues and talks with system administrators and operators. No matter how abstract a policy is, it can be studied according to these criteria. The multi-dimensional criteria for policy classification includes:

- Activity (reacting, monitoring, enforcing);
- Mode (prohibition, permission, obligation);
- Services (mail, consulting, data storage, network installation, etc.);

- Management Scenario (enterprise management, application management, systems management, network management, etc.);
- Management functionality of the policy's actions (performance, configuration, security, fault);
- Functionality of target objects (traffic management, performance analysis, security, fault location, accounting);
- Type of targets (PCs, hubs, routers, printers, employees, etc.);
- Trigger Mode (constantly active, periodic, asynchronously triggered, etc.);
- Life-time of policy (short-term, medium-term, long-term);
- Geographical criterion (global, country, city, building, department, working-group, office), and
- Organizational criterion for targets and subjects (corporate, personnel, marketing, research and development, production etc.).

Another way of specifying policy types and active policy relationships [Din00] exist. This model distinguishes between pre- and post-conditions of a policy and considers the rules of the form IF <pre-conditions> THEN <actions> WITH <post-conditions>. However, post-conditions are assumed to hold. No feedback mechanisms are provided. In turn, some well-defined policy types are provided, which may help to keep policy correlation under control. An action is a command or method invocation and it has pre-conditions, post-conditions and an activation event. A plan is a set of related actions and is used to specify dependencies between actions. The model can be summarized as follows:

```
policy ::= <policy-body> <policy-properties>
policy-body ::= IF <pre-cond> THEN {<> | <action> | <plan>}
                [ELSE {<> | <action> | <plan>} <action> | <plan>}]
                [<post-cond>]
```

Three distinct types of policies concerning policy bodies have been identified:

1) *Daemon-based policy*. This type of policy can be represented by the IF-THEN form, with post-conditions guaranteed by the action. The daemon captures the pre-conditions and triggers the action while the post-conditions are ensured by the policy according to those guaranteed by the action.
2) *Plan-Command policy*. This type of policy assumes that a goal has been given to the system and an action or plan has been appropriately established.
3) *Desirable-goal oriented policy*. In this type of policy there are no guarantees on the execution. A goal is registered as desirable, but nothing guarantees its satisfaction.

This approach allows us to specify the required policies to be active for a given policy to be applied. Even if the majority of necessary elements have been introduced, very few are found in practice.

4.1.4 Considered Policy Schema

A policy is a set of rules that have specific properties. Considering previous proposals, we represent policies as depicted in Figure 4.5 where the abstract class *PolicyFeatures* has been introduced to capture various static properties of a *PolicyGroup*.

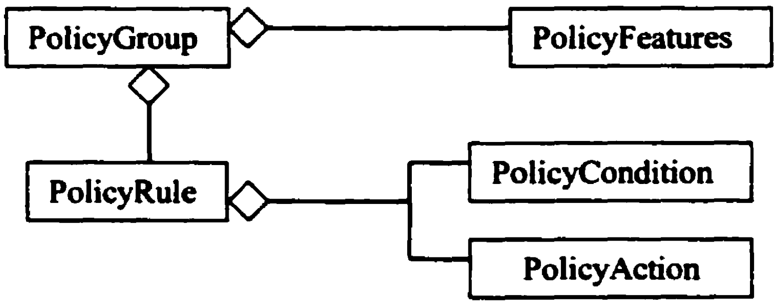


Figure 4.5 Policy Representation

The types of events and the types of interactions are represented by *PolicyActions* and *PolicyConditions*. Each different class of policies has its own policy type.

4.2 Correlation Mechanisms

Certain event models or type of interactions depend on time, others don't. In focusing on temporal issues, the following components are desired:

- an event model that takes temporal issues into account;
- operations to express the relationship in time between two events, and
- a rules model that expresses the triggering of particular actions as a function of real time.

4.2.1 Event Models with Temporal Features

An event can be represented as a pair

event = (message, time component)

where the time component may take various forms:

(a) *time component = [t1, _ ,t2]* , $t1$ = start time and $t2$ = end time.

(b) *time component = [t1, δ, _]* , $t1$ = start time and δ = event duration.

(c) *time component = [_ , δ, t2]* , δ = event duration and $t2$ = end time.

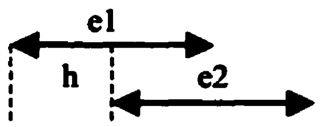
(d) *time component = [_ , δ, _]* , δ = event duration

4.2.2 Temporal Mechanisms for Processing Temporal Features

Temporal mechanisms play a critical role in monitoring network events. Knowledge about absolute and relative times of occurrence of events, their sequence and duration is mandatory. Several temporal relations [Jak95] have been defined for time-dependent event correlations. We consider the first type of event notation introduced in section 4.2.1 and define events e1 = (m1, [t1, t1`]) and e2 = (m2, [t2, t2`]). These two events may have the following temporal relations between them:

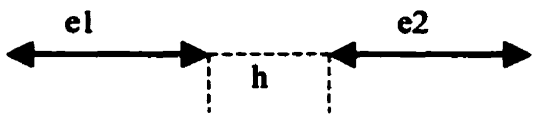
1) event e2 starts an interval h after event e1 :

e2 **after(h)** e1 <=> t2 > t1+h



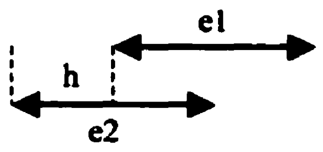
2) event e2 starts an interval h after the end of event e1 :

e2 **follows(h)** e2 <=> t2 ≥ t1`+h



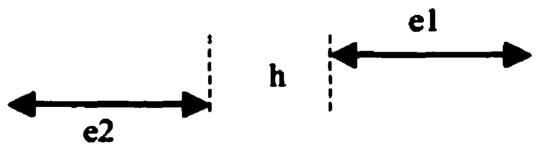
3) event e2 finishes an interval h before event e1 :

e2 **before(h)** e1 <=> t1` ≥ t2`+h



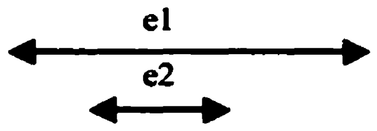
4) event e2 precedes event e2 by an interval h :

e2 **precedes(h)** e1 <=> t2 ≥ t1`+h



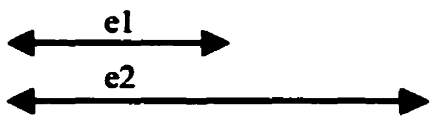
5) event e2 happens during event e2 :

e2 **during** e1 <=> t2 ≥ t1 and t1` ≥ t2`



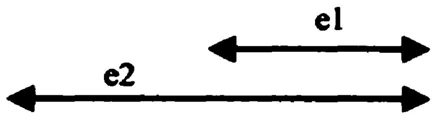
6) event e1 starts at the same time as event e2 :

e1 **starts** e2 <=> t1 = t2



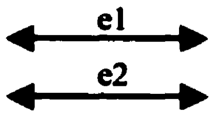
7) event e1 ends at the same time as event e2 :

e1 **finishes** e2 <=> t1` = t2`



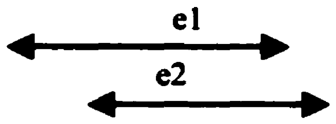
8) event e1 coincides with event e2 :

e1 **coincides** e2 <=> t1 = t2 and t1` = t2`



9) event e1 overlaps with event e2

e1 **overlaps** e2 <=> t2` ≥ t1`> t2 ≥ t1



From the above nine basic temporal relations between two events, some conclusions can be drawn:

- from definitions 1) and 2) and if d = the duration of event e1, it follows that :

  if e2 **follows(h)** e1 *then* e2 **after(d+h)** e1

- from definition 2) and 4) it follows that :

  e2 **follows(h)** e1 <=> e1 **precedes(h)** e2

- from definitions 1), 3), 5), 6) and 7), it follows that ;

  if e2 **during** e1, *then* e2 **after** e1 and e2 **before** e1 (and vice-versa)

  if e2 **after(h)** e1 and e1 **after(h)** e2, *then* e2 **starts** e2 (and vice-versa)

  if e2 **before(h)** e1 and e1 **before(h)** e2, *then* e1 **finishes** e2 (and vice-versa)

4.2.3 Events Correlation and Correlation Window

Event correlation is becoming increasingly important in managing the large volume of event messages present in today's networks. On one hand, it becomes extremely useful in treating actions prescribed by policies defined in different time zones. On the other hand, using such mechanisms, one can reduce the time of policy conflict detection and obtain network stability as a sign that a given policy has successfully applies. Among the advantages of event correlation we can mention:

- reduction of the information presented to network managers by dynamic focus monitoring an filtering;
- increase in the semantic content of the information presented to the network managers by aggregation and event generalization;
- fault isolation, diagnosis and suggested corrective actions in real-time, and
- network behavior and trend analysis.

A classical event correlation is a dynamic pattern matching over a stream of network events. Other information included in the correlation pattern includes network connectivity information, diagnostic test data, data from external databases etc.

Several results may arise after applying event correlation rules:

- a new event may be transmitted to the manager;
- an action may resolve existing events;
- a message could be sent about the existence of faults in the network;
- an executable external procedure may be called, and
- an internal system procedure may be called

All these resulting events are treated as regular events and are therefore also subject to event correlation. This creates complex, multi-level correlations.

In terms of the time model, point time and interval time are used. In point time, events take place at integer times on a pre-defined time scale. Examples of point time actions are system interrupts and user commands. In time interval, events have a time of start and the time of finish. Examples of interval time events are network elements faults and changes in network behavior.

According to the nature of the operations performed on events, several event correlation types have been identified.

1) *Compression*. Multiple occurrences of identical events are reduced to a single representative event. The number of occurrences of the event is not taken into account.

$$[a, a, \ldots, a] \Rightarrow a$$

2) *Filtering*. This operation is mainly used to reduce the number of alarms presented to the network manager. When some parameter p(a) of an alarm a does not belong to a pre-defined set of legitimates values H, then alarm a is discarded.

$$[a, p(a) < H] \Rightarrow \varnothing$$

3) *Suppression*. The event a is temporarily inhibited depending on the dynamic operational context C of the network management process. A network management context change can be due to other network events, resources, priorities etc.

$$[a, C] \Rightarrow \varnothing$$

4) *Count*. This type of correlation is counting and thresholding the number of repeated arrivals of identical events.

$$[n*a] \Rightarrow b$$

5) *Escalation*. This type of correlation assigns a higher value to a particular parameter p`(a) of event a.

$$[n*a, p(a)] \Rightarrow a, p`(a), p`>p$$

6) *Generalization*. In this correlation type, an event a is replaced by its super class b. This allows network managers to view management situations from a higher level.

$$[a, a \subset b] \Rightarrow b$$

7) *Specialization*. This is the opposite mechanism to generalization. It replaces an event by a specific subclass of this event.

$$[a, a \supset b] \Rightarrow b$$

8) *Temporal relation*. Events a and b can be correlated depending on their order and time of arrival by using this type of correlation mechanism. Different types of temporal relations have already been described in section 4.2.1

$$[a\ T\ b] \Rightarrow c$$

9) *Clustering*. Complex correlations can be created using the logical operators (and, or, not) with simple events.

$$[a,b, \ldots T, \wedge, \vee, \neg] \Rightarrow c$$

An event pending in an event list will be eliminated either by clearing by a network manager or by the expiration of its lifespan.

As depicted in Figure 4.6, each event correlation process has a *correlation time window*. A correlation time window is a maximum time interval during which the component events should take place. The length of the correlation window and the lifespan directly determine the potential of creating correlations. A wider correlation window and a longer lifespan increase the chances of creating a correlation. For fast events, the length of the correlation window should be seconds, while for slow events the correlation window can be hours or days. The right value for the correlation window and the lifespan comes from practice of managing a particular network [Jak95].

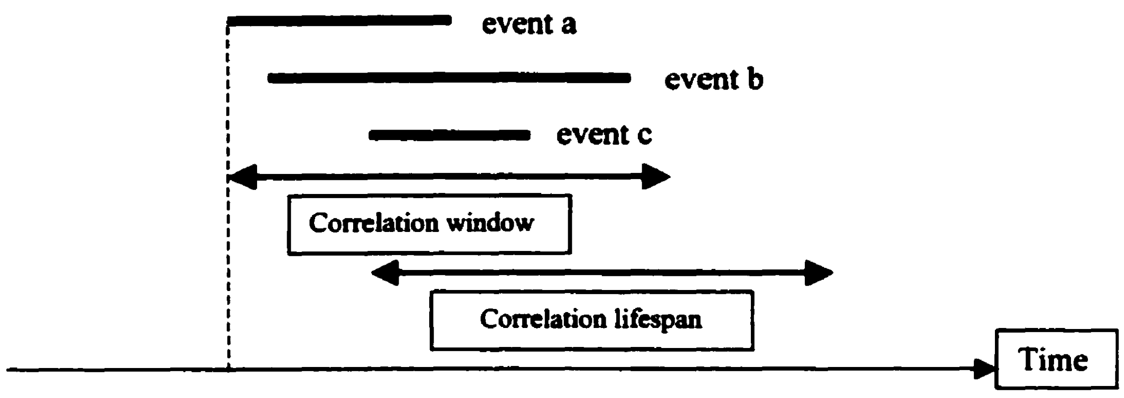


Figure 4.6 Correlation Window

4.2.4 Policy Conflict Detection

Policy conflicts can generally be classified in two categories:

(a) the first category related to policy definitions, rules (conditions and actions), etc.

(b) the second category related to temporal issues (policy execution).

(a) The first category of policy conflicts has been analyzed and discussed [Lup97]. As already defined, ***authorization policies*** specify the activities a manager or agent can perform on a set of managed objects or what monitored information can be received. Authorization policies can be *positive* (A+) or *negative* (A-). *Obligation policies* define what activities a manager or agent must or must not perform on a set of target objects. Again, obligation policies can be *positive* (O+) or *negative* (O-). Positive and negative authorization and obligation represent different policy modes. Modality conflicts are inconsistencies in policy specification, which may occur when two policies with opposite modalities refer to the same subjects, actions and targets. Three types of modality conflicts were identified:

1) **O+/O-:** the subjects are at the same time obliged and not obliged to perform the same actions on the target objects.
2) **A+/A-:** the subjects are at the same time authorized and not authorized to perform the same actions on the target objects.
3) **O+/A-:** the subjects are obliged but forbidden to perform the same actions on the target objects.

These kinds of conflicts are application specific and can not be detected directly from the policy specifications. Therefore, additional information specified in the form of meta-policies is required to detect these conflicts. Meta-policies define application-specific consistency constraints related to the contents of policies. They limit the set of acceptable attributes for policies. Different types of application specific conflicts have been identified [Mof94]: resource priority conflicts, duties conflicts, interest conflicts, multiple managers conflicts, self-management conflicts etc. These conflicts have been classified according to the overlaps between the subject, action and target sets.

In the case of modality conflicts, they arise as a result of a triple overlap between policies subjects, actions and targets. Detecting all triple overlaps between policies with opposite signs modalities will detect many possible conflicts that do not necessarily result in real conflicts. There exist several principles to establish a policy precedence relationship used as selection rule to screen out some of these possible but not actual modality conflicts:

1) *Negative policies always have priority.* A forbidden action will never be permitted.
2) *Assigning explicit priorities.* A user can specify priority values to policies. This is extremely difficult to manage and may result in inconsistencies in a distributed system in which different people are assigning policy priorities.
3) *Distance between a policy and an MO.* Higher priority is given to the policy applying to the closer class in the inheritance hierarchy when evaluating access to an object from a query. This distance between the policy and the objects to which it applies indicates the relevance of the policy and can be evaluated from the number of levels of refinement of the organizational policies.
4) *Specificity related to domain nesting.* A policy that applies to a subdomain has higher priority than a policy applying to ancestor domains. Usually, a subdomain is created for a specific management purpose, to specify a policy that differs from policies of the parent domain. Precedence based on domain nesting can be used to automatically resolve some conflicts.

(b) For the second category of policy conflicts, those related to temporal issues, it is essential that the clock system be synchronized and the absolute values of the intervals have the same origin (TimeZone):

1) There are situations in which zonal clocks might become de-synchronized and appropriate synchronization mechanisms are necessary in those cases. This issue will be further discussed in Section 4.2.5.
2) There is a zonal time delay which has to be taken into account by a translation « Δ » between time zones in order to obtain the same origin as reference. Section 4.3 will address this concern.

4.2.5 Synchronizing Zonal Clocks

A general time synchronization algorithm exists [Ber00] that analyses the time offset between any two computers' clocks in a network using mathematical topology properties. One network element, called a client, is synchronized in time with another network element, called a reference, if, at any moment, it knows the time offset between its clock and the other device's clock, or it can display the same time.

A conversion rule from a clock on host a to a clock on host b can be expressed as follows:

$$Tb(t) = Fab\,(Ta(t)) = (1+\sigma).\ Ta(t) + \delta$$

Where

σ = the shift in frequency (drift) between the two clocks.

δ = offset between two clocks at some moment

(σ and δ are not assumed to be fixed but slowly changing over time)

t = universal absolute time

Ta(t) = time showed by local clock at host a at that time

Fab( ) = conversion function that transforms time of host a in time of host b with a calculated error interval, such that the result is guaranteed and within requested maximum interval $[-\eta,\eta]$

Time synchronization is entirely handled by clients and they are responsible for guaranteeing the requested precision. To broaden the application range, the time synchronization process can be divided in two levels:

1) ***Level 1*** mechanisms are responsible for determining the offset and its uncertainty between a client clock and a reference clock. As depicted in Figure 4.7, there is a Poll mechanism that exchanges time information between the two entities involved.

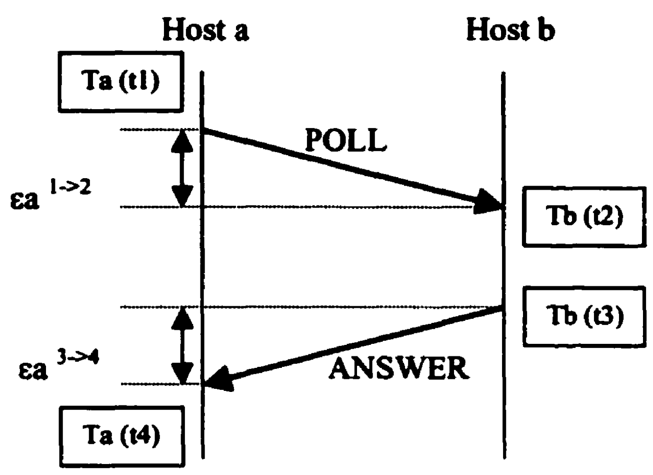


Figure 4.7 The Poll Mechanism

Through the poll mechanisms, host a collects the values of Ta (t1), Tb (t2), Tb (t3) and Tb (t4) and determines the current offset with an uncertainty which is function of the round-trip delay

$$RTa = \varepsilon a^{1\to2} + \varepsilon a^{3\to4} = (Ta\ (t4) - Ta\ (t1)) - \frac{Tb\ (t3) - Tb\ (t2)}{1+\sigma} \quad (2)$$

The triplet (ta, tb, εb) to remember in history is calculated as follows:

$$\left[\ \frac{Ta\ (t1) + Ta\ (t4)}{2},\ \frac{Tb\ (t2) + Tb\ (t3)}{2},\ \frac{RTa}{2}\,.\,(1+\sigma)\ \right]$$

2) ***Level 2*** mechanisms are responsible for calculating the conversion function and guaranteeing the required precision level. In summary, level two performs the following sequence of actions:

- obtain observation data from Level 1. If the current reference can not be found, find a new reference and restart. Else, add the new data to the history
- run the synchronization algorithm on the history to calculate the conversion function and bounding structured used to calculate errors

- calculate when a new time information should be brought in from the reference in order to ensure the requested precision.

The algorithm works to build mathematical structures on the elements of the history, which are then used to calculate the conversion function and the uncertainty interval. The history H is formed of triplets of the form $H = \{ (t_{a1}, t_{b1}, \Delta_{b1}), (t_{a2}, t_{b2}, \Delta_{b2}), \ldots , (t_{an}, t_{bn}, \Delta_{bn})\}$ which represents the current set of observation data with their uncertainty intervals. The algorithms has the following steps:

1) Calculate two sets of points, of maximum and minimum positions of observation data in their error range.
2) Build the convex closures of each one of these sets.
3) Detect crossing closures and correct them by discarding points from history.
4) Remove from history data that has not been used by at least one of the convex closures.
5) Using the convex closures, calculate infinite lines for maximum and minimum gradient for the observed clock.
6) Remove useless data from the history.
7) Choose conversion function in terms of convex closures positions and infinite lines in step 5).

The bounding structures of the algorithm limit the range of values of each parameter of the conversion function. Clock granularities allow the algorithm to get rid of errors as observation data accumulates. More information on the details of the algorithm and conclusions on its implementation are presented [Ber00].

4.3 Proposal on Temporal Aspects Considering Time Zone

Certain formalities or specification disciplines must guide the definition of a policy. According to the IETF ongoing work and some notable achievements, a policy can be represented at a high level as shown in Figure 4.5.

Although most of the policy definitions have generic, not ownership-related, characteristics, due to the large space shared by either the same company or many cooperating companies, considering temporal aspects is mandatory for correct policy definition.

Introducing temporal aspects requires a specialization of policy types (for other policies to be in place), or a new time-oriented data and action models used in policy definition.

4.3.1 Coping with TimeZone

With the advent of global market, worldwide networks are common skeletons for communicating. Implicitly, this means that the server hosting a policy set (PDP) may be widespread across many geographical areas, having a definition Time Zone.

We assume an equal division of 24 hours in a number N of time zones. Without loss of generality, let's consider the natural 24 zones used in any global temporal business reference. Therefore, we understand thereafter that a Time Zone corresponds to a «one hour band» within a 24 hours day.

We semantically distinguish between Time Zone of a PDP and Time Zone of a PEP when considering a given type of policy and one or many of its instances. Therefore, Figure 4.5 becomes Figure 4.8.

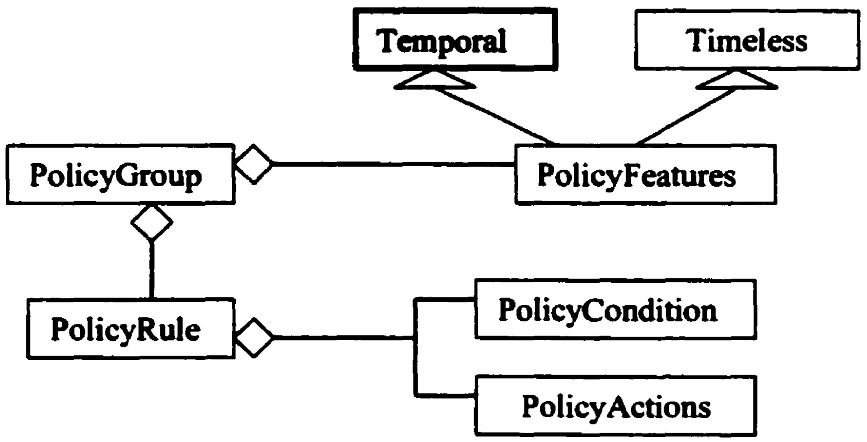


Figure 4.8 Policy Representation revised (1)

Among the desired features for a policy (Chapter 3), we introduced a category of features called "Temporal", as distinguished from those "Timeless". In the latter category we included all previous features with the exception of the temporal ones.

We define, in the Bachus-Naum Form, the data type called *TimeZone* as follows:

TimeZone :: = 1 | 2 | 3 | ... | 24

and we introduce the policy attribute

timeZone: TimeZone

(the notation a : A means that instance "a" has the type, or belongs to the class «A»)

A policy template can be represented as follows:

```
┌────────── Policy P ──────────
│
│  ----------
│
│  timeZone = 3
│
│  ----------
│
```

which means "Policy of class P is defined in time zone 3". This property will be useful in synchronizing other temporal features.

We also introduce the parameter

*applyInZone : **set of** TimeZone*

to express the fact that a given policy class, regardless of the time zone it has been defined (hosted) in, can be applied only on a given set of time zones. If it is not specified, *applyInZone* takes the same value as *timeZone* by default.

Finally, we introduce the attribute *applyingZone*, defined as follows:

applyingZone : TimeZone

with the semantics that *applyingZone* represents the current time zone an instance of a given policy is being applied in. If not specified, the attribute *applyingZone* takes the same value as *timeZone* by default.

In conclusion, we introduced and defined the semantics of the following temporal parameters:

- *timeZone : TimeZone*
- *applyInZone : set of TimeZone*
- *applyingZone : TimeZone*

where *TimeZone* : : = 1 | 2 | … | 24 and the following properties hold :

- *applyInZone* = *applyingZone*, when a policy is defined and used in the same time zone and
- *applyingZone* $\subset$ *applyInZone*, when a policy is defined in one particular time zone and used by a set of time zones.

In the present definition we do not consider the change of date effect that could easily be expressed as:

timeZone = *timeZone* + 24

We note that, according to chapter 3, Table 3.1, there exist some policy-enabled tools that do take into consideration temporal elements, but these are limited to date, day of the week or month.

4.3.2 Coping with "Companion Policies"

The notion of "companion policy" has already been needed in order to specify existential intra-policy constraints [Alp95]. However, the notion has not been extended to the policy-enabled mechanisms. We consider a new extension of the policy schema in Figure 4.8 by representing this dependency as a new type of global constraint on a policy.

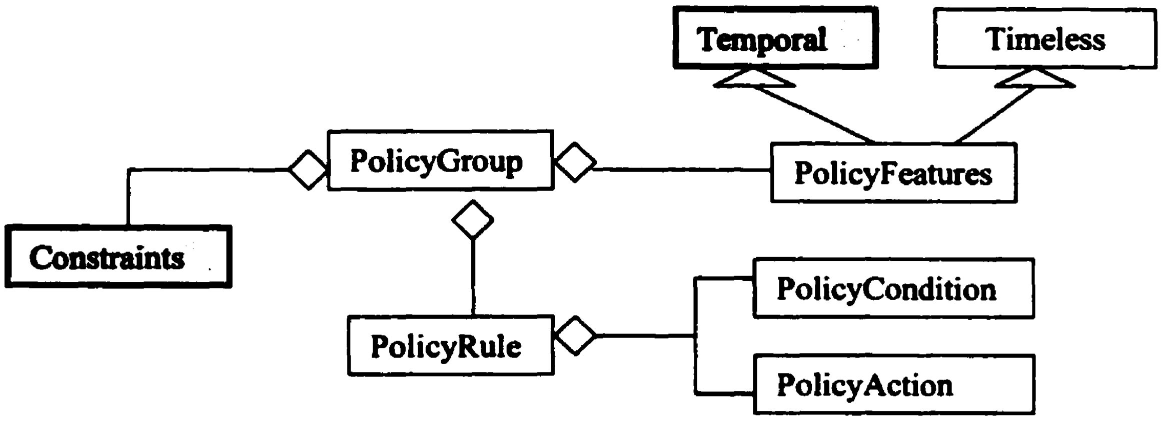


Figure 4.9 Policy Representation revised (2)

Let's consider the pseudo-operator:

require < *policyType* > **with** < *policyTypeProperties* >

where :

- *policyType* is is any of the policy types defined within the model and available to be applied (PEP) within a given set of *timeZones*.
- *polictTypeProperties* expresses particular conditions on a given set of policy instances, for example:

policyType.state = {*triggered, triggarable, blocked*}

means that the attribute *state* of an instance of *policyType* may have values within the set {*triggered, triggarable, blocked*}.

If we re-use a template-like specification, the representation of a policy can be written as follows:

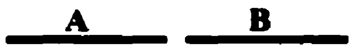
```
——— Policy A ————————
|
| ---------------
| require < policyType > with < policyType.state ={triggered, triggarable}>
| timeZone = 2
| applyInZone = 3
| applyingZone = 2
| -----------------
|
```

with the semantics
"When an instance of Policy A is necessary, then in that time zone where *applyingZone* specifies, the following condition must hold :

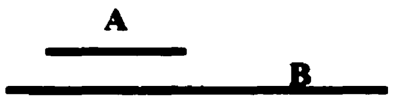
$$(policyA.applyingZone \subset policyB.applyInZone) \wedge \\ ((policyA.state = triggered) \wedge (policyB.state = triggered)) = \text{TRUE}$$ "

Obviously, the discussion is more complex if we consider some "precedence relationships" between policy states. Refined semantics may raise various case studies, for example:

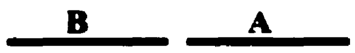
i) policy A can be triggered iff policy B can be triggered after policy A.



ii) policy A can be triggered iff policy B is already triggered.



iii) Policy A can be triggered iff policy B has been triggered and all B's actions succeeded.



This refinement is out of the scope of our proposal since it is based on the same policy model.

It is worth mentioning that Temporal Operators mentioned in Chapter 4, Section 4.2.1, can be used to express supplementary conditions between properties of policies in terms of "firing".

It is also valuable to note that, by the "start" and "end" of a policy, we understand the events outlined next. Commonly, a policy has some conditions to be fired. The last condition that determines if the logical condition set becomes true determines the firing point, or the "start". Also, if one particular condition in a set of conditions is considered as "main condition" or "activation condition", it is the one to determine the "start". Firing a policy results in a unique action or a set of actions. The "stop" point for a policy is considered to be when that unique action succeeded or failed, but terminated, or when the latest action in a set of actions (temporally speaking) succeeded or failed.

4.3.3 Coping with Relative Temporal Ordering and Dependencies between Actions

We consider, by default, that an action or set of actions resulting from a policy belonging to a given timeZone are defined with respect to the origin of the temporal interval appropriate to that time zone. For example, Zone 1 <=> [0, 1), Zone 2 <=> [1, 2), etc, where interval boundaries are hours and there are 24 hours in one day.

Sometimes, a policy having actions defined according to *policy.timeZone* may apply in another *TimeZone*.

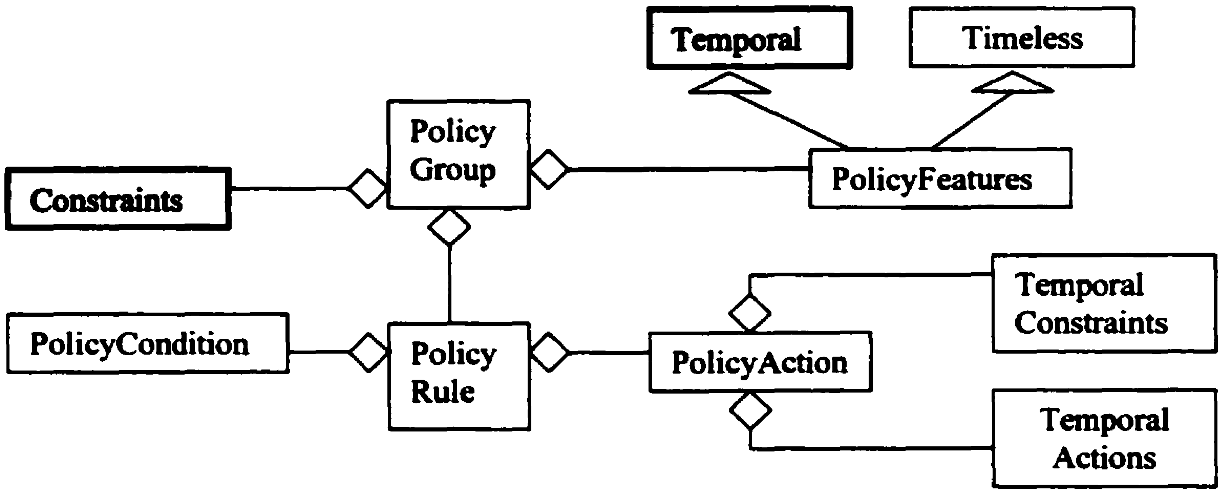


Figure 4.10 Policy Representation revised (3)

The previous generic enhanced Figure 4.9 can be revised to include the actions performed by a policy with appropriate temporal relationships.

On one hand, this allows us to properly describe the desired plan (a set of actions having the same firing conditions, but with some temporal interaction constraints).

On the other hand, one can express plans between actions having temporal constraints, but defined and possibly applied in different time zones.

Our analysis is focusing on providing the basis of the importance of considering time zones by considering the following example.

Example: Translating Temporal Events from one Time Zone to Another

Let $a_{Ti} [t_{i1}, t_{i2}]$ and

$a_{Tj} [t_{j1}, t_{j2}]$

be two actions that must be triggered by a combined policy with the following temporal constraints :

$$a_{Ti} \rightarrow \delta \rightarrow a_{Tj}$$

meaning that "a_{Ti}" will act with a delay of "δ" after "a_{Tj}" is finished.

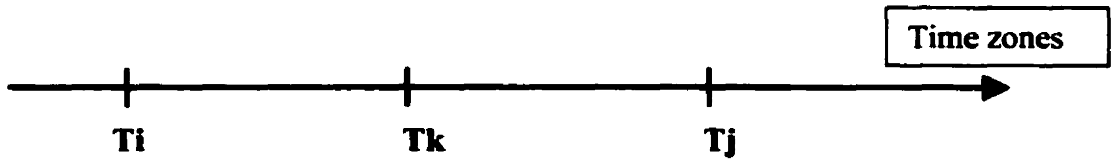
Therefore, when applied in "timeZone = k", with Ti < Tk < Tj, the following condition must apply :

$$t_{i2} + (k-i) + \delta \le t_{j1} + (k-j)$$

because a_{Ti} and a_{Tj} in timeZone k will be :

$a_{Ti} [t_{i1} + (k-i), t_{i2} + (k-i)]$

$a_{Tj} [t_{j1} + (k-j), t_{j2} + (k-j)]$



This condition automatically becomes a temporal constraint for the policy triggering a_{Ti} and a_{Tj} with δ delay.

Conclusion

We focused on policy CIM and means to provide temporal information to reflect the distribution of the networks. Our proposal extends the existing CIM and enriches the policy hierarchy with new, reusable, temporal classes.

Chapter 5

Implementation Features of the Proposal

Chapter 3 presented an extensive list of network management tools characteristics and focused on the features offered by policy-enable tolls. While very developed in areas concerning standard/IETF drafts orientation, scalability, multi-platform support, etc, many tools don't offer any support regarding temporal aspects of policy conflict detection.

Based on our tool analysis, we conclude that no temporal relationships are considered by existing policy-enabled tools, but date and day of the week. At the last IETF meeting (Adelaida, Australia, April 2000) the issue of time zones was raised and most participants recognized a need for a formal schema, eventually IETF Draft, on this topic. The present study is intended to contribute to this effort.

From the feasibility perspective, in our consideration, we looked at Sun's Bandwidth Allocator 1.0 and tried to evaluate how our proposal can enrich its set of capabilities.

5.1 Sun's Bandwidth Allocator 1.0

Before evaluating how our proposal can enrich the functionalities of Sun's Bandwidth Allocator 1.0, we present the bandwidth allocation problem in networks, a summary of Sun's bandwidth solution and its constituent building blocks.

5.1.1 The bandwidth problem

From a router's point of view, quality of service support can be broken down in three steps: definition of packet treatment classes, specification of the amount of resources (usually bandwidth) for each class of traffic, and sorting of the incoming packets into their corresponding classes. DiffServ treats the first and the third steps above, while the second issue is addressed devices called ***bandwidth brokers*** (Figure 5.1).

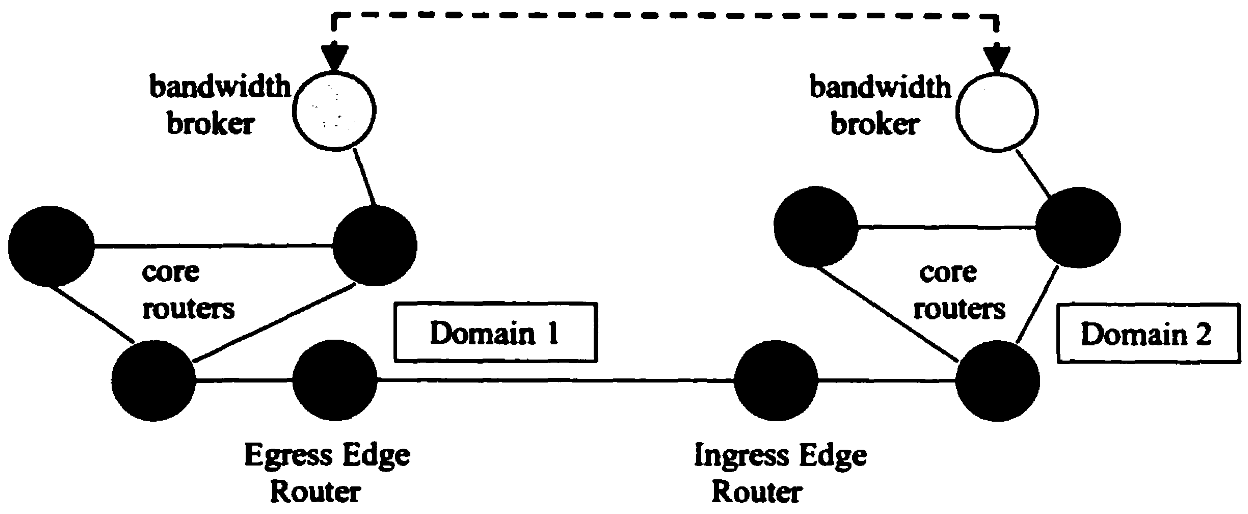


Figure 5.1 Bandwidth problem

Bandwidth brokers keep track of the current allocation of marked traffic and interpret new requests in the light of the policies and the current allocation. They are responsible for two important aspects:

- Inter-domain resource management – deals with provisioning and allocating resources at network boundaries between two domains.
- Intra-domain resource management – deals with bandwidth allocation within a network or domain. It supports information flows across a domain from an ingress point to an egress point.

5.1.2 Sun's bandwidth solution

Sun's solution to bandwidth allocation, **Bandwidth Allocator 1.0** software [Sun98], manages the bandwidth allocated to the applications, users and organizations that share the same outgoing intranet or Internet link. The two key points are (1) to prevent a small number of applications from taking up all the available bandwidth by enabling control of the allocated bandwidth and (2) to avoid congesting the network by prioritizing the traffic. The software provides guaranteed bandwidth and quality of service, monitors the levels of bandwidth and quality of service they are providing, keeps the appropriate accounts and does the capacity planning.

The outgoing traffic is managed based on the type of traffic (FTP, e-mail, telnet, etc) and on the end-user or organization source or destination address. Any type of TCP or UDP-based traffic can be managed by the Sun Bandwidth Allocator. It works in a heterogeneous environment without any modification of the systems accessing the gateway and can run on top of WAN links (PPP) or LAN links (Ethernet and FDDI).

The Packet Classifier component collects packets from the IP layer, applies the filters defined in the provisioning rules and assigns each of them to its appropriate class queue where it waits to be processed. If the queue to where a packet is allocated is full, the packet is dropped. Its retransmission will ensure that the packet is resent.

The following factors are important when assigning a packet to a particular class:

(1) Protocol (TCP or UDP)

(2) IP source address

(3) IP destination address

(4) TCP or UDP source port

(5) TCP or UDP destination port

The configuration of Sun's Bandwidth Allocator 1.0 specifies, in terms of some or all of the above factors, the set of known classes for a network node. It also allocates a priority and a percentage of bandwidth to each class. Classes are arranged hierarchically and every class has a parent.

The Packet Scheduler component decides the order in which class queues are processed based on the percentage of bandwidth configured and the priority of each class. Within a queue, the FIFO (first in first out) priority is applied. When the network traffic reaches the maximum allocated to a class, packets from the next lower priority class are processed.

Sun Bandwidth Allocator 1.0 can be installed in two environments: as a Traffic Manager or as an Application Controller.

(1) As traffic manager

When installed in the "IP-transparent" mode of operation on a LAN, WAN or Internet server, the SUN Bandwidth Allocator 1.0 can control the outgoing traffic and still remains transparent to the IP users. In this mode of operation, any failure of the hardware or software can make the link unavailable. One of several solutions can be used to ensure higher availability, namely to reduce the path redundancy with intelligent routing algorithms such as OSPF, to reduce router redundancy with Hot Standby Routing Protocol combined with a routing protocol which converges very rapidly (such as EIGRP) or to link redundancy with link monitoring by fault tolerant switches.

(2) As application controller

When installed as an Application Controller, SUN's bandwidth allocation software controls output from a server to a LAN, WAN or Internet. The server can be a file, an application or a Web server. For example, traffic outgoing from a cluster of Web servers of an ISP can be prioritized according to server or application. In this manner, the ISP can guarantee bandwidth to each customer, depending on the hosting contract.

Sun Bandwidth Allocator 1.0 gives real-time statistic information including: class id, class bandwidth, traffic volume, number of dropped packets etc.

Statistics can be accessed directly via Java Graphical User Interface or via any SNMP manager such as Solstice Domain Manager or Solstice Site Manager. A statistics API allows the building of user-defined monitoring utilities or the integration of Sun Bandwidth Allocator 1.0 statistics into the user's own monitoring system.

Accounting is done as depicted schematically in the Figure 5.2 below.

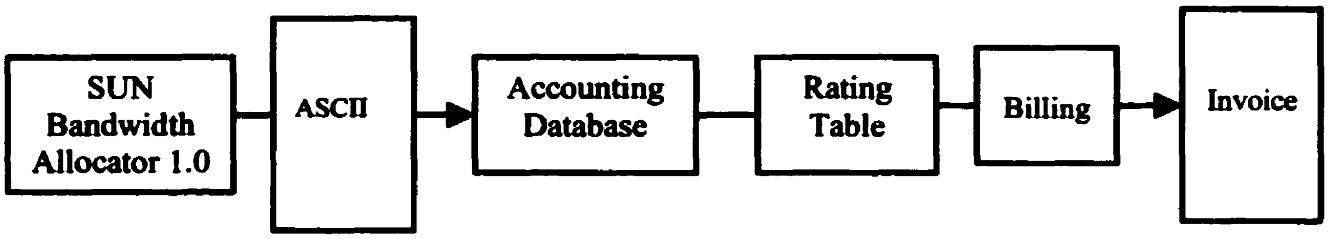


Figure 5.2 Sun Bandwidth Allocator 1.0 Accounting System

An accounting agent of the SUN Bandwidth Allocator 1.0 software collects accounting information and outputs it in ASCII format which eventually is accepted by the Billing system. The accounting information includes sender/receiver IP address, port numbers, packet and byte counts, time-stamps. The accounting data also allows for the billing of advanced, quality-of-service-enabled IP services such as premium class-of-service traffic.

5.1.3 Building Blocks

Sun's Bandwidth Allocator 1.0 has at its base the STREAM module found between the IP layer and the network interface. Figure 5.3 below shows a schematic overview of the building blocks of the software, which, as seen in Chapter 3, is a general management tool.

A new building block can be attached to the Sun Bandwidth Allocator building blocks schema to deal with temporal issues of policy conflicts. The resulting tool will be temporal policy-enabled. This enrichment of the tool implies:

- considering a new block implementing the issues related to time zones, called "Temporal Building Block".
- updating "notification" messages with appropriate alerts for handling temporal constraint violations.

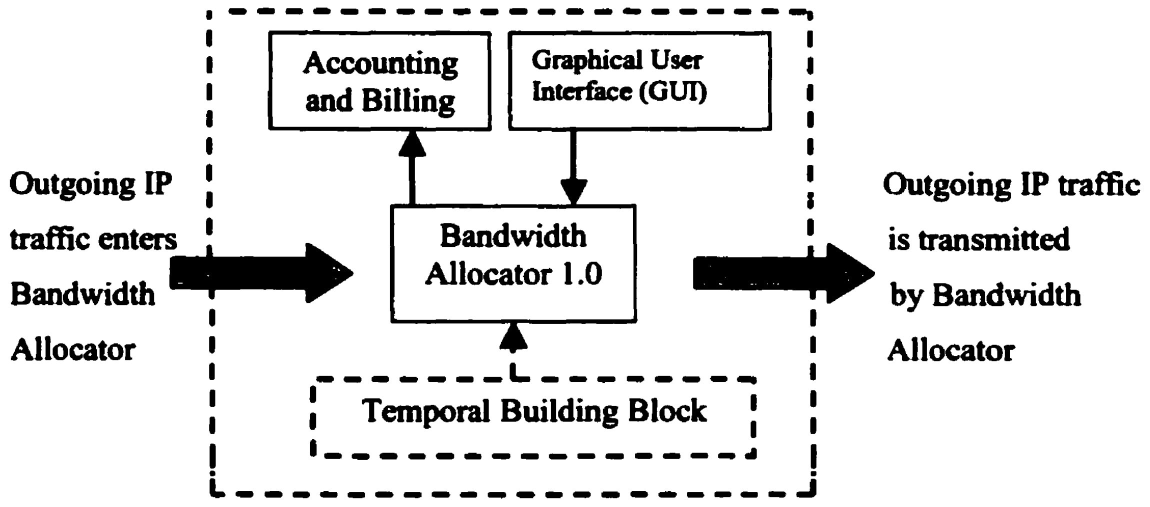

Accounting and Billing
Graphical User Interface (GUI)
Outgoing IP traffic enters Bandwidth Allocator
Bandwidth Allocator 1.0
Outgoing IP traffic is transmitted by Bandwidth Allocator
Temporal Building Block

Figure 5.3 Bandwidth Allocator building blocks

5.2 Temporal–Oriented Building Block

The classes of the policy CIM are intended to serve us as an extensible class hierarchy, through specialization, for defining policy objects that enable application developers and policy administrators to derive easily temporal aspects and detect functional conflicts.

The main classes that must be used, according to the CIM schema, are presented in Chapter 2 (Figure 2.6):

- *policy (abstract)*
- *policyGroup (structural)*
- *policyRule (structural)*
- *policyCondition (auxiliary)*
  - *policyTimePeriodCondition (auxiliary)*
  - *vendorPolicyCondition (auxiliary)*
- *policyAction (auxiliary)*
  - *vendorPolicyAction (auxiliary)*

with the following attributes :

Class: policy (type: abstract), Attributes:

- cn (commonName, optional)
- caption (optional)
- description (optional)
- policyKeywords (optional)

Class: policyRule (type: structural), Attributes:

- policyRuleName (required)
- policyRuleEnabled (optional)
- policyRuleConditionListType (optional)
- policyRuleConditionList (optional)
- policyRuleActionList (optional)
- policyRuleValidityPeriodList (optional)
- policyRuleUsage (optional)
- policyRulePriority (optional)
- policyRuleMandatory (optional)
- policyRuleSequencedActions (optional)

Class: policyCondition (type: auxiliary), Attributes:

- policyConditionName (required)

Subclass: policyTimePeriodCondition (type: auxiliary), Attributes:

- ptpConditionTime (optional)
- ptpConditionMonthOfYearMask (optional)
- ptpConditionDayOfMonthMask (optional)
- ptpConditionDayOfWeekMask (optional)
- ptpConditionTimeOfDayMask (optional)
- ptpConditionTimeZone (optional)

Subclass: vendorPolicyCondition (type: auxiliary), Attributes:

- vendorPolicyConstraintData (optional)
- vendorPolicyConstraintEncoding (optional)

Class: policyAction (type: auxiliary), Attributes:
- policyActionName (required)

Class: vendorPolicyAction (type: auxiliary), Attributes:
- vendorPolicyActionData (required)
- vendorPolicyActionEncoding (required)

In order to specialize and aggregate various temporal policy-related classes, we have introduced, we consider the last Internet Draft on Policy CIM, version 1, May 2000.

In the following, we will situate the new classes we have proposed within the IETF hierarchy. IETF introduced the notion of *LocalTime* and *UTCTime* in *PolicyTimePeriodCondition.* An instance of *PolicyTimePeriodCondition* has five properties that represent times: TimePeriod, MonthOfYearMask, DayOfMonthMask, DayOfWeekMask, and TimeOfDayMask. All of these properties represent one of two types of times: (i) local time at the place where the policy is applied or (ii) UTC time. The property *LocalOrUtcTime* indicates which time representation is to be applied to an instance of PolicyTimePeriodCondition. Because PCIM only provides for local and UTC time, a policy management tool that provides other time representations needs to map from these other representations to either local or UTC time.

Mostly, IETF temporal properties concern policy validity in terms of policy specification. Our input is mainly focusing on policy execution and conflict detection at the execution time.

We considered *PolicyFeature* as an abstract « attribute-based » class and specified two concrete classes, i.e. *Timeless* and *Temporal.*

As an example of *Timeless*, we consider IETF Proposal of the policy property called "Priority". The "Priority" property gives a non-negative integer for prioritizing policy rules relative to each other. A larger integer value signifies higher priority. One of the goals of this property is to allow specific policy rules to temporarily override established policy rules. This translates into the fact that instances having this property set, have a higher priority that all instances that lack this property. This prioritization among policy rules provides a mechanism to resolve policy conflicts.

Our *Temporal* class may contain all the newly introduced attributes, i.e. *timeZone*, *applyInZone*, and *applyingZone* and other IETF-defined properties such as *TimePeriod*, which refers to the validity of a policy definition. This property identifies an overall range of calendar date and times over which a policy rule is valid.

By considering our proposal, the policy root will include the following supplementary classes:

- *policyFeature*
  - *Timeless*
  - *Temporal*
- *policyConstraints (companion)*

We conclude that the new policy hierarchy allows a policy-based tool to express execution constraints coping with TimeZones, which is a novelty.

Chapter 6

Conclusions and Future Work

Systems management represents the set of activities necessary to ensure that information systems function according to user requirements and objectives [Mof94]. Commonly, large networks have a number of types of hardware/software with specific features. These features differ from component to component and cover many capabilities related to capacity, scalability, speed of connection etc. In order to manage these components, they are modeled by so-called managed objects (MOs).

We have seen that several problems present in large-scale management systems have been identified [Slo94] and that mechanisms simplifying management are proposed [Put95] [Mas93].

Policy driven management systems are proposed to cope with policy changes. The desired mechanisms provided by a policy management service include (i) the creation, modification, and deletion of policies, (ii) the representation and interpretation of policies, (iii) the storage and retrieval of policies, (iv) the negotiation of the outcomes of policy conflicts and (v) the communication of new policies or modifications done to existing policies to managers.

Today, Quality of Service (QoS), Type of Service (ToS), Class of Service (CoS) mechanisms, such as Differentiated Services (DiffServ), as well as Integrated Services (IntServ) are at the base of policy-based network management and prioritizing resources to requirements.

The ongoing work on IETF Policy CIM tries to standardize a common hierarchy Of policy features to facilitate policy reuse and policy-based tool interaction.

Some of the temporal issues concerning policy validity (definition) have already been proposed and mainly adopted by the emerging policy-enabled tools. However, temporal properties dealing with execution are from far solved. We proposed certain solutions for aspects concerning the notion of time zones. We enhanced the current model with appropriate classes of required information. More specifically, our solution copes with:

- companion policies;
- time zones , and
- start/stop translation from actions specified by different policies defined in different time zones.

Future work that is under consideration refers to:

- policy notification;
- sensors and measurement for confirming the correctness of firing for translated actions, and
- run-time policy conflict handling.

For the first topic, the work on typing appropriate events must be developed according to COPS primitives and data structures prescribed by IETF Policy CIM.

The second topic requires new mechanisms for feedback to avoid the network collapse if a policy fails or succeeds only partially.

The third becomes a complex task. We estimate that dedicated "conflict detection mechanisms" embedding temporal issues for run-time solutions must arrive as a concern from many vendors and industrial research groups.

Bibliography

[All00] NetPolicy -Data Sheet, Allor Communications, 2000
http://www.allot.com/products/Policy_Manager_DS.htm

[Alp95] B. Alpes, H. Plansky, "Concepts and Applications of Policy-Based Management", Integrated Network Management IV, Edited by A.S.Sethi, Y.Raynaud, F.Faure-Vincent, Chapman&Hall, IFIP 1995. P. 58-68

[Ban00] «Network Bandwidth Management – Bandwidth Management Demo», 2000
http://www.sun.com/software/bandwidth/demo/DemoBM.html

[BER] OSI "Basic Encoding Rules" for [ASN.1], ISO standard 8825, December 1987

[Ber99] Y. Bernet, R. Yavatkar, P.Ford, F. Baker, L.Zhang, K. Nichols, M.Speer, R. Braden, Interoperativity of RSVP/IntServ and DiffServ Networks, February 1999, draft-ietf-diffserv-rsvp-02.txt, Work in Progress

[Ber00] Jean-Marc Berthaud, "Time Synchronization Over Networks Using Convex Closures", IEEE/ACM Transactions on Networking, April 2000, Vol.8, N.2, IEANEP (ISSN 1063-6692)

[Bir96] Alessandro Birolini, "On the Use of Stochastic Process in Modeling Reliability Problems", Springer-Verlag, 1985 (Lecture Notes in Economics and Mathematical Systems, 252)

[Cci91] CCITT, Common Management Information Service Definition, ITU-T, International Standard, 1991.

[Cis00] Cisco QoS Policy Manager, Product Documentation
http://www.precept.com/warp/public/cc/cisco/mkt/enm/cap/qospm/index.shtml

[Coo00] Mike Cookish, "COPS lays down policy-management law", Network World Fusion News, www.nwfusion.com/news/tech/0510tech.html?nf

[Con99] Policy-Based Network Management, by Joel Conover, November 29, 1999
http://www.networkcomputing.com/1024/1024f1side1.html

[Cor89] Thomas H. Cormen, Charles E. Leiserson, "Introduction to Algorithms", ISBN 0-262-03141-89 MIT Press.

[Cui95] Cui-Qing Yang and Alapati V.S. Reddy, "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks", IEEE Network Magazine, July/August 1995, V.9, n.5

[Der96] Luca Deri, "Surfin'Network Resources across the Web", IEEE Second International Workshop on Systems Management, June 19-21, 1996, Toronto, Ontario, Canada.

[Din95] Petre Dini, Gregor v. Bochman, "Modeling for Automatic Policy-Driven Reconfiguration within Distributed Systems", IGLOO Technical Report, CRIM 1995

[Din00] P. Dini and N. Mihai, "Policy Framework: Specifying Policies Types and Active Policy Relationship", Policy Workshop, Bristol, UK, 2000

[Dme92] Open Software Foundation, OSF Distributed Management Environment (DME) Architecture, 1992

[Dmtf] DMTF, Common Information Model (CIM) version .1, DMTF September 1997.

[Ege96] K. Egevang, P. Francis "The IP Network Address Translator (NAT)", 1996 Network Working Group, rfc 1631
http://194.52.182.96/rfc/rfc1631.htm

[Fes99] O.Festor, P.Festor, N.Ben Youssef and Laurent Andrey, "Integration of WBEM-based Management Agents in the OSI Framework", Integrated Network Management IV, Edited by Morris Sloman, Subrata Mazumdar and Emil Lupu (1999)

[Fir99] "PORTUS Firewall Tutorial", 1999
http://www.lsli.com/tut27.html

[Fla99] J.P. Martin-Flatin, "Push vs. Pull in Web-Based Network Management", Integrated Network Management VI, Edited by Morris Sloman, Subrata Mazumdar and Emil Lupu (1999)

[Ful00] Bob Full, Mike Roderick, Michele Clark, Jeff Vian "Radius--Remote Authentication Dial In User Service"
http://www.squashduck.com/~roundman/radius/

[Haf98] A. Hafid and G.v. Bochman, « Quality of Service Adaptation in Distributed Multimedia Applications », Multimedia Systems, 1998, v.6, n.5, p.6, 22p.

[Hei99] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, « Assured Forwarding PHB Group », RFC 2597, June 1999

[Hew00] OpenView PolicyXpert Frequently Asked Questions, 2000
http://www.openview.hp.com/docs/199.htm

[Hou95] Houck, K., Calo, S., Finkel, A. 1995. Towards a practical Alarm Correlation System. In Integrated Netwotk Management IV, eds. A.S. Sethi, Y. Raynaoud, F. Faure-Vincent, Chapman&Hall, 1995.

[How97] Timothy A. Howes and Mark C. Smith, "LDAP- Programming Directory-Enabled Applications with Lightweight Directory Access Protocol", 1997

[Hut94] D. Hutchison, G. Coulson, A. Campbell and G.S.Blair, « Quality of Service Management in Distributed Systems », Lancaster University Report, Number MPG-94-02.

[Int00] Differentiated Services Moving towards Quality of Service on the Ethernet http://www.intel.be/network/white_papers/diff_serv/index.htm

[Ips00] "IP Security and NAT: Oil and Water?", 2000 http://www.isp-planet.com/technology/nat_ipsec_p2.html

[Iso1] ISO, Structure of Management Information – Part 4 : Guidelines for the Definition of Managed Objects, International Standard 10165-4, 1992.

[Iso2] ISO, Structure of Management Information – Part 7 : General Relationship Model, International Standard 10165-7, 1995.

[Iso3] ISO, Specification of Abstract Syntax Notation Number One (ASN.1), ISO, International Standard 8824, 1990.

[Iso7498] "Information Processing Systems – Open Systems Interconnection – Basic Reference Model – Part 2 : Security Architecture" , IS 7498-2 , ISO/IEC , 1988

[Iso10164] "Information Technology- Open Systems Interconnection –Systems Management – Management Functions", IS 10164-X, ISO/IEC

[Iso19] ISO/IEC CD 10164-19, Management Domain and Management Policy, Management Function, January 21, 1994

[Itu] ITU-T, Principles for a Telecommunications management network, ITU-T, International Standard M. 3010, January 1996.

[Jac99] V. Jacobson, K. Nichols, K. Poduri, "An Expedited Forwarding PHB", RFC 2598, June 1999

[Jak95] G. Jakobson, M. Weissman "Real-time telecommunication network management : extending event correlation with temporal constraints", Integrated Network Management IV, Edited by Adarshpal S.Sethi, Yves Raynaud and Fabienne Faure-Vincent, 1995

[Jon99] G. Jones, E. Zeisler, L. Chen, "Web-based Messaging Management Using Java Servlets", Integrated Network Management IV, Edited by Morris Sloman, Subrata Mazumdar and Emil Lupu (1999)

[Jor93] J.F Jordaan and M.E. Paterok "Event Correlation in Heterogenous Networks Using the OSI Management Framework", Integrated Network Management, vol. III, 1993

[Kat97] S. Katker, M. Paterok, "Fault Isolation and Event Correlation for Integrated Fault Management", Integrated Network Management V, Integrated Management in a Virtual World, Edited by Aurel Lazar, Roberto Saracco and Rolf Stadler, 1997

[Koc96] Thomas Kock, Christoph Krell and Bernd Kramer, "Policy Definition Language for Automated Management of Distributed Systems", IEEE International Workshop on Systems Management, June 19-21, 1996, Toronto, Canada

[Koc95] T. Koch, B. Kramer "Towards a Comprehensive Distributed Systems Management", ICODP'95, 20-24 February 1995, Brisbane, Australia,

[Luc99] Lucent Technologies announces RealNet Rules policy management application, April 1999
http://www.lucent.com/press/0499/990426.cob.html

[Lup97] E. Lupu, M. Sloman, "Conflict Analysis for Management Policies", Integrated Network Mangement V, Edited by Aurel Lazar, Roberto Saracco, and Rolf Stadler, 1997.

[Lup97a] E. Lupu and M. Sloman. "Towards a Role-based Framework for Distributed Systems Management" Journal of Network and Systems Management, vol. 5, no. 1, pp. 5-30, Plenum Press Publishing, 1997.
http://www-dse.doc.ic.ac.uk/~ecl1/

[Lup97b] E.C. Lupu and M.S. Sloman "A Policy Based Role Object Model", First International Enterprise Distributed Object Computing Workshop (EDOC'97), Gold Coast, Queensland, Australia, pp. 36-47, Oct. 1997.
http://www-dse.doc.ic.ac.uk/~ecl1/

[Mac93] M. Masullo and S. Calo, "Policy Management : An Architecture and Approach", In [IWSM-I 93]

[Man97] A. Mankin, et al, « Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement – Some Guidelines on Deployment », RFC 2208, September 1997

[Mar98] L.P. Martin-Flatin "IP network management platforms before the Web" Technical report SSC/1998/021, version 2, SSC, EPFL, Lausanne, Switzerland, December 1998.

[Mas93] Masullo MJ, Calo SB, "Policy Management: An architecture and Approach", Proceedings of the IEEE First International Workshop on Systems Management, Los Angeles (1993)

[Mcb91] McBrian P, Niazette M, Pantaziz D, Selviet AH, Sundin U, Theodoulis B, Tziallas G, Wohed R., A Rule Language to Capture and Model Business Policy Specifications, Proceedings of the Third International Conference on CaiSE, Norway (1991)

[Mof94] J.Moffret and M. Sloman "Policy Conflict Analysis in Distributed System Management", Ablex Publishing Journal of Organizational Computing, 4 (1), 1-22

[Nat99] Cisco IOS Network Address Translation (NAT) http://www.cisco.com/warp/public/701/60.html

[Ngu92] Incorporating Business Management Policy into Information Technology: Nguyen TN. Proceedings of the Second International Symposium on Network Management, Halfway House, South Africa (1993)

[Ngu93] Thang Nguyen , "Linking business strategies and IT operations for Systems Management Problem Solving", In [IWSM-I 93]

[Nou97] Mhamed Nour, Abdelhakim Hafid, J.William Atwood "Routing with Quality of Service Constrains" , to appear in International Pacific Workshop on Distributed Multimedia Systems, 1997, Vancouver, Canada

[Oli95] C. Olivera, J. Kim, and T. Suda, "Quality-of-Service Guarantee in High-Speed Multimedia Wireless Networks", http://jblevins.ics.uci.edu/Dienst/UI/2.0/Describe/ncstrl.uci%2fICS-TR-95-41

[Omg95] Object Management Group, "The Common Object Request Broker: Architecture and Specification ", 1995, OMG Document PTC/96-08-04

[Omg921] "Object Management Architecture Guide", Document 92-11-1, Object Management Group, September 1992.

[Omg922] "Object Services Architecture", Document 92-8-4, Object Management Group, August 1992.

[Orc99] Orchestream Enterprise Edition 2.0, Fighting network decay - Take control of network performance,1999. http://www.actualit.com/products/orchestream/enterprise.htm

[Osi91] Open Systems Interconnection : Management Overview. IS10040 (1991)

[Pel93] A. Pell, C. Goh, P. Mellor, J-J. Moreau, S. Towers, Data + Understanding = Management, in W.W. Chu, A. Finkel (eds) : Proceedings of the IEEE First International Workshop on Systems Management, Los Angeles, IEEE, April 1993.

[Poo91] Poo CD, "Representing Business Policies in the Jackson System Development Method", The Computer Journal Vol 34 no 2 (1991)

[Por97] C. Pornavalai, G. Chakraborty, and N. Shiratori, « Routing with QoS Constraints in Integrated Services Networks », IEEE Conference — Multimedia Networking PROMS-MmNet'97

[Pro98] George Prodan, "Extreme Networks Introduces the ExtremeWare Enterprise Manager Policy-Based Management Platform", 1998 http://www.extremenetworks.com/corporate/pressroom/news/pr23.asp

[Put95] Towards policy driven systems management by Phillip Putter, Judy Bishop and Jan Roos, Integrated Network Management IV, edited by Adarshpal S.Sethi, Yves Raynaud and Fabienne Faure-Vincent (1995)

[Raj99a] R. Rajan, S. Kamat, P. Bhattacharya, D. Biswas, "Networking Policy Condition Information Model", April 1999, <draft-rajan-policy-conditions-00.txt>, Work in Progress

[Raj99b] R. Rajan, S. Kamat, J.C. Martin, M. See, R. Chaudhury, D. Verma, G. Powers, R.Yavatkar, "Policy Action Classes for Differentiated Services and Integrated Services", April 1999, <draft-rajan-policy-qosschema-01.txt>, Work in Progress

[Rei99] F. Reichmeyer, K. H. Chan, D. Durham, R. Yavatkar, S. Gai, K. McGloughrie, S. Herzog, A. Smith, "COPS Usage for Policy Provisioning", February 1999, <draft-sgai-cops-provisioning-00.txt>, Work in Progress

[Sab97] B. Sabata and S. Chatterjee, « Taxonomy for QoS Specifications », Proceedings of WORDS '97, Newport Beach, California, 1997 , February 5-7

[Sak99] Sakir Yucel, Nikos Anerousis, "Event Aggregation and Distribution in Web-based Management Systems", Integrated Network Management VI, Edited by Morris Sloman, Subrata Mazumdar and Emil Lupu (1999)

[Sal84] J. Saltzer, D.Reed, D. Clark, End-to-End arguments in System Design, ACM Transactions in Computer Systems, November 1984 www.reed.com/Papers/EndtoEnd.html

[Sch97] A. Schade, "An Event Framework for CORBA-Based Monitoring and Managemet Systems", Open Distributed Processings and Distributed Platforms, Edited by Jerome Rolia, Jacob Slonim and John Botsford, 1997

[Sim94] W. Simson, "The Point-to-Point Protocol (PPP)", 1994
ftp://ftp.isi.edu/in-notes/rfc1661.txt

[Slo94] Sloman MS, Twidle K., "Domains : A Framework for Structuring Management Policy" Chapter 17 of Network and Distributed Systems Management. Sloman MS, Kappel K. Addison Wesley (1994)

[Sni00] Y. Snir, Y. Ramberg, J. Strassner, R.Cohen, "Policy Framework Internet Draft", QoS Policy Schema, expires September 2000,
draft-ietf-policy-qos-schema-01.txt

[Sni00a] Y. Snir, Y Ramberg, J. Strassner, R. Cohen "QoS Policy Information model", Internet draft , draft-ietf-policy-qos-info-model-01.txt

[Sni00b] Y. Snir, Y. Ramberg, J. Strassner, R.Cohen, "QoS Policy Schema",
Internet Draft <draft-ietf-policy-qos-schema-01.txt>
http://www.ietf.org/internet-drafts/draft-ietf-policy-qos-schema-01.txt

[St399] Introduction to QoS policies – white paper, Stardust.com.Inc, 1999,
www.stardust.com

[St199] QoS protocols and architectures—white paper, Stardust.com Inc., 1999
www.stardust.com

[St299] Quality of Service Glossary of Terms, Stardust.com Inc., 1999
www.stardust.com

[Sta93] William Stallings, SNMP, SNMPv2 and CMIP, The Practical Guide to Network – Management Standards, Addison-Wesley Publishing Company, Inc., Chapter One, 1993

[Sta99] Introduction to QoS policies, stardust.com
http://www.winsock2.com/policy/whitepapers/qospol.htm

[Ste00] Steve Steinke, Simple Network Management Protocol—SNMP,
http://www.networkmagazine.com/magazine/tutorial/management/9702tut.htm

[Str99] John Strassner, "Directory Enabled Networks", 1999

[Str00] J. Strassner, E. Ellesson, B. Moore, "Policy Framework Core Information Model", Internet Draft <draft-ietf-policy-core-info-model-06.txt>

[Str00b] J. Strassner, E. Ellesson, B. Moore, Ryan Moats, "Policy Framework LDAP Core Schema", draft-ietf-policy-core-schema-06.txt, November 04, 1999

[Str99] John Strassner, "Directory Enabled Networks", 1999

[Sun98] Bandwidth Management for IP Networks Sun Bandwidth Allocator, Version 1.0
http://www.sun.com/software/white-papers/wp-sba10/

[Sun95] SUN Microsystem. EmbeddedJava. Available at :
http://www.javasoft.com/products/embeddedjava/

[Terp92] Terplan, K. 1992. Communication Network Management, Practice-Hall Inc. , 1992

[Tiv97] "Tivoli Remote Control", 1997
http://www.tivoli.com/products/index/remote_control/overview.html

[Tow99] W. Townsley, A.Valencia, A. Rubens, G. Pall, G. Zorn and B. Palter "Layer Two Tunneling Protocol L2TP", 1999
ftp://ftp.isi.edu/in-notes/rfc2661.txt

[Vol98] L.C Wolf, C. Griwodz, and R. Steinmetz, "Multimedia Communication", IEEE Network, 1998, v.12, n.6, p.56, 8p.

[Wah00] Lightweight Directory Access Protocol (LDAP) , Chair(s): M. Wahl Mark.Wahl@innosoft.com, Applications Area Director(s): Ned Freed ned.freed@innosoft.com and Patrik Faltstrom <paf@swip.net>
http://www.ietf.org/html.charters/ldapext-charter.html

[Wel94] C. Wells, "Tivoli Systems Inc. , TME", Datapro Integrated Network Management, January 1994

[Wie95] Rene Wies, "Using a Classification of Management Policies for Policy Specification and Policy Transformation", Integrated Network Management IV, Edited by Adarshpal S. Sethi, Yves Raynaud and Fabienne Faure-Vincent

[Yav99] R. Yavatkar, D. Pendarakis, R. Guerin, "A Framework for Policy-based Admission Control", April 1999, <draft-ietf-rap-framework-03.txt>, Work in Progress