Chapter 3

Proposal of a policy-based management architecture

In this section, the design of the management platform and the divers items that take part to support the QoS requirements imposed by both the final users and the service providers are detailed.

The management platform scenario proposed was developed as a part of the research project Network Management and Intelligence in the Internet 2 (GIRIN)[1], which studies different management mechanisms for advanced networks as the Internet 2. Some of the references where this network scenario is proposed are: [Reyes00-1], [Reyes00-2]

[1] This Project has the support of the CICYT TIC 2000-1042

The implementation environment is based on a differentiated service scheme in which, in order to guarantee the QoS requirements that different applications demand today, we have designed four different classes of service that are explained in detail in [Reyes01-1].

The proposed system is based on a network with differentiated services that offers routing with QoS restrictions. The system allows the fact of giving priority to some specific kinds of services or users and the application in the network of high-level business policies that are important in a certain moment. In order to achieve that, all the items in the network are configured, managed and controlled by means of policies pre-defined by the network administrator and by means of policies that the system creates itself. The application of such policies depends on several factors as for example, service and user's priority levels, the resources availability, the QoS required for a specific service, etc.

The system proposed keeps its scalability aggregating the traffic flows in the four classes of service that were proposed to reach the different application requirements.

One of the most interesting aspects in this project is the great amount of state-of-the-art technologies that are involved, for example, the architecture CORBA is used to enable the management in the network and its Internet communication protocol Inter ORB Protocol (IIOP) [IIOP] is used to define the interfaces that support the interoperability and the transparency to build applications distributed into platforms from different manufacturers, both hardware and software manufacturers. In the same way, several protocols are also involved, as for example, the Multi-protocol Label Switching (MPLS) [Rosen01] and the Lightweight Directory Access Protocol (LDAP) [Hodges02]. Several technologies as for example, monitoring agents, databases, etc. are used as well.

The following diagram shows the systems involved and the existing interconnection among them:

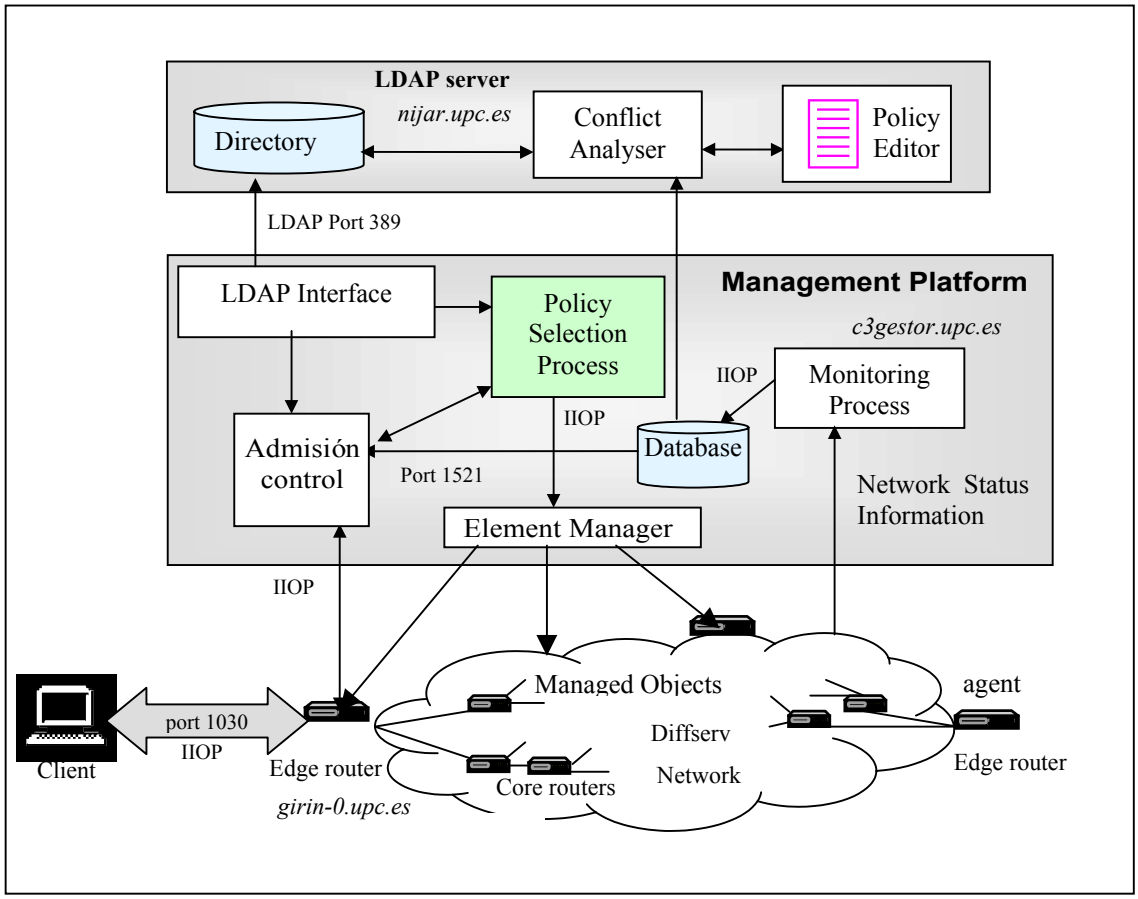

Figure 1. Systems involved into the GIRIN scenario

The policy key aspect, a different one from the fact of having a high-level declarative nature, is that policies can be seen as vehicles to force the management systems functionalism because they allow them to evolve when the requirements change. Therefore, a policy-based system must provide flexibility to add, change or eliminate management intelligence, while in case of traditional management models; the management logic owns a static nature, parameterised only by means of the managed objects actions and attributes.

3.1 General Functioning of the system

This system functioning will be explained taking as a start point the instant in which the user requires to be connected to the system. The user's terminal sends the connection parameters required to the edge router, which transfers the request to the policy management platform. The first step to accept or deny a connection is made by the admission control module, which actives an identification process and security mechanisms as for example, the user's authentication. Once the user has been authenticated, the SLA corresponding to the user is searched in the database. In this agreement a paragraph is about the SLS profile associated with the user's SLA. All parameters used to define a SLS profile are explained in section 3.6. Those parameters mark the value rank that a connection can use, that is to say, if the parameters that the user's connection requires are out of this rank, the connection cannot be accepted.

The edge router sends the management platform both, the execution parameters and the source and destination address that the user requires. These addresses must be conformed with the scope parameter belonging to the SLS profile that the user owns.

In the management platform, one of the main modules is the policy system, which analyses the user's requirement and the available resources in the network to determine the policy that has to be applied in the edge router by means of the use of the access control module and the provisioning one. These modules only use two policy actions, one that accepts the user's connection and one that denies it.

In case the connection is accepted, the provisioning system sends the service request parameters to the policy evaluator to determine which is the suitable policy or whole of policies that must be applied in the network items both to satisfy the user's request and to keep a desirable state in the network. The policy actions chosen by the policy evaluator indicate the edge router the first hop that must be done, the corresponding QoS and a possible sequence of the following hops, that is to say, the necessary whole of labels MPLS to route the package. In the same way, policies indicate the configuration that core routers

through which the package will pass must have. See chapter 5 and for more information about path selectors for MPLS there are interesting references in [Anjali02], and about policies for MPLS in [Brunner01] .

After the first hop, in case there are no problems in the network, then the pre-established hop sequence is continued. If a problem appears, then the policy evaluator must be consulted about the action that must be carried out to come the package routing to an end.
In the system proposed, the policy application points (PEPs) are found in the network items and are responsible for the policies application at real time while the applications in the system are executed. In order to that, the policy server, once it has consulted the repository where the policies are stored, must lower the corresponding objects to create the specific policies that must be applied.

When a new policy is stored in the repository, a notification is sent to the policy evaluator to indicate the place where the new policy was stored and the PEPs to which it affects. In this way, the evaluator will always have the updated information about the policies that each PEP can apply in the different network items.

The core routers and edge routers configuration is defined by a scripts generator in charge of defining a file with the specific logic to reconfigure each device in the network being based on the policies selected by the policy evaluator. Such scripts configure, obtain information and invoke available operations both in the local objects and in the remote objects using the communication protocol IIOP of CORBA [IIOP].

The automatic script generation is obtained from the high-level policy representations stored in the repository and by means of a translation process that maps those representations in a language that the differentiated service network devices used in this Thesis can understand.

3.2 Management system components

The policy management tool consists of the following components.

3.2.1 Monitoring process

It is necessary to consider a tool providing the characteristics of all network devices and its connections automatically, together with the application information and the users operating in the network at a certain moment, in order to obtain a consistent policy application.

Monitoring process considers different types of events in the network, for example, the alarms that emerge when a network item fails or when more calls than a specific router expected to give full satisfaction arrive. In the same way, the monitoring process collects information from the edge routers, the core routers, pre-established links, etc.

The monitoring process is based on Java Agents. To go deeper in this work, the reference [Reyes00-3] can be consulted. The events and the network information are stored dynamically in the database in order to make the policy evaluation process own this information as available and updated to apply the corresponding policies correctly.

The core routers monitoring consists of obtaining the four basic parameters that define a class of service: bandwidth, delay, jitter and losses. We have carried out some investigations related to the obtaining of these four parameters in every network node in the references [Reyes00-2] [Barba02]. In the Appendix II all monitoring process implementation details are provided together with the description of the existing routes and the QoS levels corresponding to every route.

3.2.2 Routers based on differentiated services.

Our network scheme is based on a differentiated service and MPLS network. Section 5 gives a more detailed explanation about the routing performance. This section shows how we configure the edge and core routers based on policies over a Differetiated Services Technology.

The core routers have the function of adding the information fluxes, doing the DS classification and sending back the packages; edge routers have several functions depending on the considered type of router. In this project we differentiate 3 different types of edge routers basically.

a) Fist-hop router: It is the closest router considering the package sender host. The packages fluxes are classified and marked according to the SLS profile assigned to the connection. It is responsible for the fact of establishing an agreement between the traffic and the bandwidth that the user and the service supplier have considered.
b) Ingress router: It is located on the ingress points of the DiffServ domain and it carries out the classification of all packages based on DS field.
c) Egress router: It is located on the egress points of the DiffServ networks to control the traffic. It carries out the classification of all packages based on DS field.

Scheme 2 shows the necessary steps to configure the DiffServ routers that we use. There are three processes: traffic control, routing and exit process. The implementation of these modules is detailed in the Appendix II.

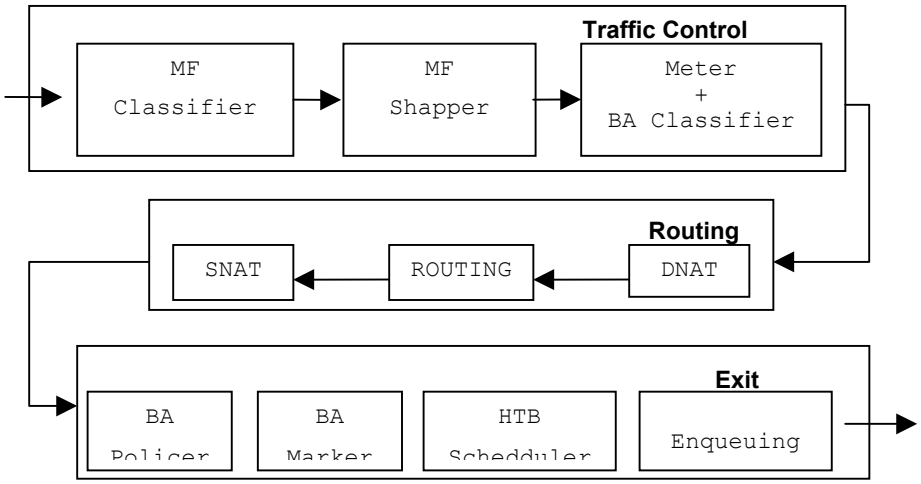

Figure 2. Steps to configure the Diffserv Routers

3.2.3 Policy Enforcement Points (PEPs)

PEPs are in charge of the policy or set of policies application selected by the policy selector, that is to say, they are in charge of configuring the network elements.

There is a process between the policy selector and PEPs that is in charge of translating the high-level policies into low-level ones, so that PEPs can receive the policy action that must implement translated into an order that DiffServ routers can understand.

PEPs change depending on the network due to the fact they must use the specific parameters in every device to be able to configure them. In this management platform the following routers parameters are configured by: classifiers, meters, markers, queues and schedulers. However, due to the fact that the proposed system can be adapted to heterogeneous networks, in case only one different type of network is used, for example, an ATM network, only the parameters that the policy translator produces must be changed and instead of configuring the classifiers, meters, etc. the parameters corresponding to the chosen network will be configured.

3.2.4 Provisioning process

This process has several functions: it carries out some basic security functions, as for example, the user's identification and authentication. We can also mention that these security functions can be as complex as we want them to be, simply by implementing the corresponding security policies. Once all security processes have been done, the provisioning process obtains, from a user's request, the execution parameters that the connection needs, which are sent to the admission control module.

Provisioning process accedes to the database that stores every user's profile and it is there where the SLS profiles assigned to the user are recuperated. Those SLSs depend on the agreement that the user has established with the service supplier. This information is also sent to the admission control module.

The last of the points that the provisioning system must send to the admission control is a report including the network alarm state and the general link state in a specific moment. In order to achieve that aim, the provisioning module has access to the database in which the monitoring system stores the network state dynamically.

3.2.5 Policy-based Admission Control.

Admission control is in charge of deciding if a user's connection request based on the policies belonging to the admission role previously stored in the LDAP directory is accepted or denied. There is an IETF specification about how a Framework for Policy-based Admission Control works [Yavatkar00]. In this Thesis, Admission control carries out some operations based on policies to know if the network can guarantee enough resources to a new connection. Our admission Control also revises the alarm report, if there are some congestion or block alarms, then it sends the request to the policy selector to determine what to do, and if it is pertinent, then Admission Control accepts the connection.

Admission control verifies that the requested destination address is in the valid source-destination addresses rank that the SLS profile scope parameter (corresponding to the connection) indicates. In case the source or destination address is out of the permissible rank, then the connection is rejected.

Afterwards, all required execution parameters for the connection along with the SLSs profiles to which the user has access are analysed. Depending on the QoS level that the connection required by the user needs, one of the SLSs profiles is chosen. In case any of the SLSs profiles satisfies the connection, then the user's request is denied or a best effort is provided. SLSs profiles establish the parameter ranks through which a service can be released. The fact these requested parameters are out of these ranks indicates that the network topology or technology cannot satisfy some specific execution levels (as for example, very low or very high levels) or that the requested parameters rank was not previously established in the service level agreement.

In a QoS network all connections are provided only if the network is able to guarantee that it will keep the service level in all execution parameters from the entry node to the exit node. Having as a basis the monitoring report that the provisioning system sends to the admission control module, we have to determine if there are enough available resources to be able to guarantee the requested execution levels for the new connection. When the network is unable to guarantee the requested service with the requested quality, then the policy server determines which action to take in case it is necessary to provide this service.

This can happen, for example, if the requested service has extremely priority. Otherwise, there can also be policies that offer a different Quality of Service to the user in order to release the service.

Policies on which the admission control module is based are recuperated directly from the LDAP directory. We can specify several policy roles, for example one role for SLS policies, another role for marketing policies, network ones, admission ones, etc. One

example of admission control polices could indicate that a specific service cannot be provided during a specific time period.

There are only two possible policy actions that the admission control can apply: the acceptation action and the reject action in relation to a connection. This decision is sent to the provisioning process in charge of sending the actions to the PEPs. If the connection is accepted, then the provisioning process sends a routing way request and the network state report to the policy evaluation process.

3.2.6 Policy editor

This module is extremely important to achieve one system to be stable, reliable and scalable. The network administrator uses the policy editor to create the high-level policies that will control all the system components. The policy edition process can use imperative programming languages such as C, C++, Java, XML, or specialised languages for the policy definition that provide some degree of management abstract. Upon this abstract, administrator can specify the desired behaviour of management system by policies.

Next, there is a description of some approaches to specify policies:

Logic-based policy definition language (LPDL). This language is based on first order logic and adopts Prolog's syntax. An administrator can use LPDL to define a set of reaction rules whose format is (<Event>, <Reaction>). Messages to a policy server can come from management modules, which are responsible for monitoring and controlling network's running, or from other policy servers. In response to these messages, the policy server generates appropriate communication events, then searches local reaction rules set, analyses the state repository and may generate an internal event, send messages to management modules and/or other policy servers, and record state changes into state repository in order to handle with these communication events. The syntax of reaction rules encompasses two types of events, communication and internal event, and three types of actions, "data action"

to manipulate state repository, “event action” to generate an internal event and “do action” to generate a message to be sent to management module or other policy server. [Li01]

Policy Description Language (PDL). PDL, [Koch96] was developed in the University of Munich to specify the actions that must be executed after receiving an event. Lobo in [Lobo99] uses PDL as a programming language of the policy-based network management system developed in [Virmani00].

PDL allows the description of policies to configure the network nodes infrastructure and to guarantee the access to the different resources. One of the PDL language advantages is the fact that it does not allow the event generation by means of a policy application. However, the policy description is not separated from the event specification, it means that every policy must introduce the name of the event that uses. PDL does not allow defining new events due to the fact that all events are received from previously defined agents and it does not consider an event description language.

The event specification as a part of the policy description consists of a name and their attributes, followed by the keyword DELIVERS. The attributes are limited to variables such as “string, long and boolean”. Thanks to the help of variables, it is possible to reuse the attribute values within the policy specification in the part of ACTION and CONTRAINT of the policy. See table 1.

POLICY name FOR subject ON ¡target¿ DELIVERS event-string-long-boolean attribute”

Table 1. PDL Tags

Security Policy Language (SPL). SPL comes from the Institute Mageland in Portugal [Ribeiro99]. This language was designed to express policies that help to decide about the event acceptation. This acceptation of every event depends on its qualities (for example,

author, destination, action, etc.) of the specific context in which all events happen and it also depends on the qualities of past events. SPL entities (users, files, objects, events, etc.) own an interface by means of which all qualities can be consulted.

A SPL policy consists of wholes and rules; the first ones include the entities that policies use to decide about the event acceptation and the second ones are event functions that can have three different classes of values: deny, allow and do not apply. Rules are also classified into simple ones and compound ones. Simple rules consist of a binary expression to decide which is the domain for the rule application and to define the acceptation of the applicable events to the rule. Rules can be composed of other rules and policies through the use of ternary algebra. In these cases, every policy must own a rule called "consulting rule" (identified by the question mark), which is a whole of other rules and defines the policy behaviour. Delegation is obtained when an instance to a policy is made and it is used as a rule in the composition of other rules. The main SPL disadvantage is the fact that it is not suitable in the use of all management platforms.

Ponder. One of the older and advanced free-distribution policy languages is the Ponder [Dulay01], which was developed by the Imperial College of Science in England. It is a declarative language pointing to objects to specify management policies and security ones to distributed systems. The language allows to cover the wide range of the current requirements that the paradigms in the distributed systems present using domains to make easier the specification of the policies related to big-scale systems. Policies are specified to objects collections that are stored in domains instead of individual objects, which provides scalability and flexibility.

Ponder supports basic policies, compound ones and meta-policies.

Basic policies can be:

1. Authorisation policies. They define the access control and the privilege use over the destination objects.

2. Obligation policies. They specify which actions must be executed over fixed objects when specific events happen.
3. Delegation policies. They specify actions that a user can delegate to others.

Compound policies consist of:

1. Groups. They are policies and restrictions that own similar semantic relations and must be used together.
2. Roles. They specify rights and obligations for organisational positions.
3. Relations. Among policies that define roles.
4. Classes of specialisation. They allow the extension to new policy definitions using heritage.

There are other less-developed languages, for example, the CacheL [Fritz99], which is a language used to specify policies in general. There are also some researches on policy languages such as XML, which is a hypertext language that can be used to represent high-level policies.

However, despite this language variety of free-distribution policies, in this Thesis we chose to implement policies using the Java programming language, due to the fact that those previously mentioned languages present some disadvantages, as for example: The Ponder main disadvantage is its high level of complexity caused by a great number of constructors. The Ponder compiler only generates the implementation rules. That is to say, the mentioned rules still need mapping manually to the specific application code. Other languages are too specific to represent policies from different functional areas and some other languages require complicated compilers to achieve the storage of the policies in the repository.

The policy edition is carried out taking into account the PCIM model [Moore01] of the IETF presented in section 2.1.4 of this Thesis, which supposes great advantages in the moment of storing policies in the LDAP directory.

Edition conflict analyser. The edition module includes a conflict analyser that avoids the edition of new policies that can bring instability in the network, ambiguity in the policy selection or direct conflicts with other policies previously defined and stored in the LDAP directory, that is to say, policies in which the same condition produces a different action cannot be admitted. The solution before policies that are not congruent is to eliminate the previous policies or to change the new policy definition.

On the other hand, the edition conflict analyser must control that the value rank assigned to the policies is within the limits that the network establishes either by means of topology or by technological capacities and within the limits that the network administrator establishes. For example, the conflict analyser does not let the storage of a policy that indicates the premium CoS assignation to a router that is not able to support the corresponding parameters to this kind of service.

3.3 Distribution policy mechanisms

There are several mechanisms to distribute the low-level policies in the different network elements. The IETF proposes that the distribution of configuration has to be carried out by storing the low-level policies in a LDAP directory to make the servers of every domain recuperate the corresponding policies and distribute them to their network elements. However, the distribution of the configuration can also be carried out as follows:

1. Using a configuration file that a policy agent being executed in the system can read. The network will be configured according to this file. When a configuration file is copied in a known location, for example, /etc/pagent.conf, the system makes a configuration of itself automatically according to the specified rules.
2. Translating the low-level policies into the configuration files that every device needs and copying them in the router or in the adequate server remotely.
3. Translating the low-level policies into automatised scripts for the specific routers. The entries attributes are provided by means of several commands options.

4. The policy distribution using management frameworks is a solution for the necessity of managing different devices from a central location. Some examples of these frameworks are: TME-10 framework that uses Tivoli systems to their management and TNG framework by Computer Associates. Every framework is typically developed in an environment with a manager console and several agents. These agents are located on the devices being managed and they are responsible for the application of directives coming from the management console. They can send reports, alarms or any other alert to the console too.

In the proposed network scenario, the standard architecture oriented to objects for applications (CORBA) defines the interfaces to support the interoperability and transparency to build distributed applications among different platforms and to distribute policies between the different network elements.

Corba uses the IIOP communication protocol to store the events originating in the network items in a database. In [Reyes00-1], [Reyes00-2] and [Reyes01-1] the design can be observed and the Appendix II shows the implementation of the mentioned application, where we mainly use CORBA interfaces related to the Quality of Service and routing.

3.4 Policy levels

Policies define the management strategy in a system and according to them the network behaviour is determined. The abstraction level in a policy covers from really high-level business aims to low-level policies aims that are configurable in the network devices. For example, if a company wants to establish specific execution levels for its communications network, the execution aims would be the high-level policies. These aims become specific assignation rates for the different traffic flows. Assignation rates are the low-level policies that will be applied in the different network elements.

There are different opinions about the number of policy levels. Verna [Verna00] proposes four levels of policies:

1. Business level policies. An administrator introduces this kind of policies.
2. Network level policies. They are defined in a format that allows communicating the network policies among the different devices within the network.
3. Device level policies. They are specified in a generic format that carries out a mapping of the exact configuration of a specific device.
4. Device configuration. It is the exact configuration of a specific device.

The business level policies describe the business process in a higher level than in the network level policies and the difference between the last two levels is mainly syntactic, it means that in the device level, policies are represented independently from any other entity and can be applied to more than one device. The description and automatization in these policies levels is still an open problem for the research community.

In the Imperial College of Science they work with two basic policy levels: the authorisation policies and obligation policies [Sloman94].

In this Thesis a policy scheme based on three basic levels is used: high-level policies (service policy level), medium-level policies (network level policies) and low-level policies (device configuration level policies).

1. High-level policies (service policy level). They define the network administrator aims. These policies are defined in intuitive formats for human operators. The high-level policy specification is independent from the underlying network technology, for example Business goals, special offers for a specific kind of service or a specific kind of user.
2. Medium-level policies (network level policies). These policies are defined closer to the network than from the user. Policies from network level are useful for Admission Control process, routing, etc. It is possible to make a mapping between high-level policies into network-level policies, it is necessary to

consider the Classes of Service that the network provides and to establish the high level sentence in a clause "If…. Then …" The group of actions is limited depending of the policy role, for example in case of admission control policies, the action could be accept or not accept a service.

3. Low-level policies. These policies must establish in detail the exact operations that a network device has to execute, for example, in a Differentiated Services network, policies define the traffic mapping to the different classes of packages marking.

3.4.1 Conversion of service level policies into network level policies

Every service is defined considering some network requirements. PBMS can be used in order to make use of this service. The case we are considering now also uses the help of intelligent agents that have to be applied in order to provide the management of some network resources. In this sense, the use of a service requires a network management according to this service requirement.

Parameters defined by services allow the construction of service management policies. Parameters defined by services have a correlation with parameters able to be monitored from the network and the management of the corresponding network resources that can be executed with it.

The relation among the different classes of service with descriptor parameters defined from those ones is described in the paragraph 3.5 of this chapter. Management carried out with the network resources follows the aim of accomplishing the specifications to provide an adequate quality for the services.

The mapping between the network management and the corresponding service management is carried out for every role defined in the PBMS following similar criteria to the way defined in TMN.

A typical case is the provision of voice services over the IP network. The management of these services requires a gatekeeper (Rec. H.323 of ITU-T) that manages the necessary bandwidth in every user's session. In case of the PBMS architecture proposed here, the bandwidth being managed is substituted by a service quality that can be defined from the metrics: retard, losses and bandwidth. This aspect allows us to accomplish the rule H.323 requirements with the best-effort Internet network and it also allows a higher integration with mobile communication networks where losses and retard are more restrictive.

3.4.2 Conversion of network level policies into devices configuration

In order to translate the network level policies into policies that can be implemented in the network items, we can consider four stages:

1. Names mapping, in which the high-level constructors like users names and applications or groups of users and applications are mapped to field headings as IP addresses and ports numbers.
2. Service-kind mapping. The definition of the network level service classes is changed to specific technology parameters. In the field of operations used in this Thesis, a service-kind mapping towards the necessary parameters in a Differentiated Services network has been carried out.
3. Priority determination. The relevance of a policy for a device or whole of devices is examined here.
4. Grouping, where all devices with applicable identical policy wholes are gathered in a common structure. The IETF proposes that the devices with similar functions can have the same role in the network and as a consequence, the grouping process can be considered as the role determination that a device owns.[Moore01]

Every policy specifies a customer, an address, an application and a kind of service. The mapping rules must also specify the kind of service that will be assigned in case of a congestion in the network, as for example, when a packet cannot be mapped within the kind of service that usually maps it. This situation takes place when the traffic rate assigned to a specific kind is exceeded in a device and as a result, that kind of service is not able to accept more packages.

Examples

As an example, see the Appendix II.4 about Router Agents. In that place a description explains how the agents apply over device configuration by means ob managed beans. The interaction of intelligent agents with the database allows the introduction of monitoring information in the system.

In another example, the conversion of network-level polices into the device configuration depends on the implementation carried out by every manufacturer. The system allows the integration of LDAP directories with SNMP architectures. The agent implementation and the private MIB database are required for SNMP.

3.5 Classes of Services mapping (CoS)

The Quality of Service is not provided to all services or all users in the same way. Because of that, some classes of services are established to determine the convenient quality parameters depending on the different kinds of traffic. The policies that the network administrator defines are in charge of the device configuration based on the parameters of the classes of service. In this way, each traffic type is mapped towards a class of service, see table 3.

The ITU-T 1,356 recommendation specifies the use of three service classes for the ATM networks. The class 1 guarantees the most strict requirements of CBR traffic; the class 2 is used for applications that do not differentiate losses between CLP=0 and CLP=1; the class 3 is used for applications that expect a guaranteed execution level for cells with CLP=0, but they do not expect guarantees in cells with CLP=1.

CTD 400ms (class1)
CDV 10-8 quintiles 3ms (class 1)
CLR=1 3+10-7(1) 10-5(class2)
CLR=0 10-5(class 3)
CER 4+10-6 (all)
SECBR 10-4 (all)

The ABR category provides access to users that can adjust their transmission rate. In exchange for this user's co-operation, the network provides a very low-loss service. The application specifies a maximum PCR transmission rate and a minimum MCR rate. This service does not provide any limits in the jitter, and so, the applications executed at real time are not good candidates for ABR. Some examples of applications for ABR are LAN interconnection, high-execution file transference, non-sensitive traffic to the jitter, etc.

The UBR service has no delay restrictions or jitter, and so, the ATM forum compares this service with the traditional IP Best Effort. This traffic is considered as risky due to the fact that the network does not provide any execution guarantee. Some examples of applications are LAN emulation, IP over ATM and all non-critic mission traffic.
The service nrtVBR category supports applications without any restrictions about the delay or jitter, but owns a variable rate. This service supports applications with blasts of traffic and expects a low CLR. These applications include data package transference and file transference.

The rtVBR is used for time sensitive applications; that is to say, they have delay and jitter restrictions. They transmit a restricted-time variable rate to a PCR and to an average rate defined by the SCR and the MBS. Some examples of these applications are voice, a variable-bit rate video, etc.

CBR supports applications at real time with a bandwidth fix capacity defined by the PCR. It admits strict variations restrictions in delays. Some examples of these applications are voice, video in constant-bit rate and the circuit emulation service. In general, the CBR service is useful for those applications that require a dedicated capacity and the minimum jitter.

The nrtVBR service category works for the commuted packages data applications that are sensitive to losses such as frame relay, IP and LAN traffic.

On the other hand, networks based on differentiated services have defined two kinds of Per Hop Behaviours (PHB), Assured and Expedited Forwarding. As the name indicates, PHB means the behaviour in each hop. However it is also possible to define the behaviour in a domain, [Nichols01] details how to create Per Domain Behaviours using some rules for their specification. Next, we mention the already specified PHBs that our system uses, but the system can use new PHBs or new PDBs.

Assured Forwarding [Heinanen99] has four classes, each one with three precedence levels. In each DiffServ node, the AF class reserves a certain quantity of resources (space in the buffer and band width). In each AF class, IP packages are marked by means of one out of three precedence possible values. In case of congestion, high-loss precedence packages are first discarded.

Expedited Forwarding [Davie02] is used to implement premium service or a virtual link. It is useful for those applications that require low delay, low jitter, low losses and a bandwidth assured through DiffServ domains. The service level agreement specifies the

maximum bit-rate for a specific flux or for aggregated fluxes, being the user the only responsible of avoiding exceeding the established bit-rate. In case he or she exceeds it, then the exceeding traffic can be eliminated or delayed. The ISP guarantees that the bandwidth is available. The network administrators are then in charge of guaranteeing that this type of traffic is not the owner of all the bandwidth and it leaves some space for the assured service and the best effort.

In this Thesis, a policy-based scheme is used in order to keep the required bandwidth limited quantity in each application. This mentioned scheme allows the possibility of dynamically reassign the established flow for every class. In this policy system, a mapping of the main classes of service defined for the ATM networks and for the differentiated services is carried out using four classes of service presented in detail in the following tables.

ATM Service types	**Typical Uses**
Constant Bit Rate(CBR)	Real-time, QoS guarantees
Real-Time Variable Bit Rate(rt-VBR)	Statistical multiplex
Non-Real-Time Variable Bit Rate(nrt-VBR)	Statistical multiplex
Available Bit Rate(ABR)	Resource Exploitation, feedback control
Unspecified Bit Rate(UBR)	Best effort, no guarantees

Table 2. ATM Service Types

Parameters		"Alfa" Class EF, CBR	"Beta" Class AF,ABR,GFR	"Delta" Class DRT, rtVBR, nrtVBR	"Gamma" Class *Best effort* VBR
Guaranteed Bandwidth		Sí	Opcional	Si	Si
Adequate for Burst			Si	Sí	
Adequate for real time				Sí	
Voice service	Bandwidth	64 Kbps	64 Kbps	12-24 Kbps	
	Delay	<200ms	<200 ms		
	Jitter	<48ms	<48 ms		
	CLR	<19 ms $=10^{-3}$			
Video Service	Bandwidth	140 Mbps	140 Mbps	1-5Mbps	
	Delay	<200 ms	<200 ms	<200 ms	
	Jitter	Para buffer receptor de 4Mbps=10ms	10 ms	10 ms	
	CLR	$7.6 * 10^{-12}$		10^{-9}	
Data Service	Bandwidth			0.1-1Mbps	50 Kbps
	Delay			1000 ms	
	Jitter			100 ms	
	CLR	$3*10^{-7}$	10^{-5}	10^{-6}	

Table 3.Mapping of CoS [Reyes01-1]

Alfa Class was designed to fulfil the most rigorous requirements of ATM's CBR traffic and EF traffic from differentiated services. In general, Alfa supports real time applications that require a fixed capacity defined by the maximum rate of transmission, therefore, the network needs to assign it the maximum rate of transmission to Alfa service. This type of service supports strict restrictions with delay variation. Examples of application that can be transmitted through Alfa are voice, constant-bit video, and circuit emulation services.

Beta Class transmits applications that do not differentiate loss between CLP=0 and CLP=1. This class of service works with sources that can change their transmittal rate. Beta provides dynamic access to current capacity (which is not in use by other types of services), and enables users to adjust their transmittal rate. Beta service does not provide delay variation limits, making real time applications not suitable for it. Examples of Beta services are LAN´s interconnections, high performance files, unrestricted traffic, etc.

Delta class of service is devoted to applications that suppose guaranteed performance in cells with CLP=0 but does not wait for guaranteed cells with CLP=1. This class of service supports time sensible applications that have restrictions regarding delay requirements and jitter, though have a restricted maximum and minimum variable time rate. This type of service can support gust traffic and a lower loss rate. This type of service functions with data applications sensitive to losses, such as: frame relay, IP, and LAN traffic. In addition, it can be used with delay variation applications such as voice and high variable rate video.

Gamma services do not have any delay or variation restrictions, and do not provide specific or guaranteed quality of service. Therefore, this type of traffic faces risk because the network does not offer any kind of guarantees regarding performance. Gamma is not the best for any application, but is the most economical. Local area networks and the Internet are examples of this type of best effort service, along with LAN, and IP over ATM.

3.6 Creation of SLS Profiles

The Quality of Service is offered according to agreements between an ISP and its users or between several ISPs. This agreement is a formal definition of the services that a client requires and those that the Internet provider can actually offer. The SLS parameters simplify such agreements very much, due to the fact that they describe services without any ambiguity. Therefore, it is very important to consider a SLS that allows high levels of automating.

We proposed characterising a connection based on the Service Level Specification corresponding to each user. This mentioned specification is determined depending on the Service Level Agreement (SLA) that the user and the network establish. Nowadays, SLS parameters are not standardised, but there are several research projects related to the parameters production and the application of specifications in the service level.

The proposed management platform uses three SLS profiles, which represent a great advantage for users who provide all technical resources to the QoS provider in a dynamic way.

SLS profiles must provide the policy system the necessary information to execute the admission control and configure the network nodes to offer a specified service. However, nowadays there is not any standardised SLS. There are researching projects related to the production of parameters, algorithms and protocols to let negotiation, monitoring and the service level specification application. The project TEQUILA (Traffic Engineering for QUality of Service in the Internet, at LArge Scale) [Goderis00] is one of the most developed and accepted researches.

The SLS profiles that we created have eight parameters; seven are from the TEQUILA project [Goderis00-1]. Following, it is a briefly description of the parameters.

Scope. Indicates where the QoS policy for that specific service offering is to be enforced. In this way, scope only identifies the topological/geographical region upon which a specific quality of service must be applied indicating the limits of this region clearly. Scope has to be specified by a couple of ingress and egress addresses.
There are several policies using scope parameter, for example, if we want low-priority users not to have access to routing paths in which a specific server is or to deny the access to a specific group of servers or sub networks. Following the last examples, a way to block the access is to indicate as invalid these addresses in the scope parameter.

Flow Description. The flow descriptor in a SLS provides the necessary information to classify the packages in a DiffServ edge node indicating the IP packages that belong to a specific service and on which a QoS policy will be applied to let this classification be done.

An identification flow groups the IP datagrams that share at least one of the following characteristics: code point of the DiffServ (DSCP), information from the destination, the source or the application. Therefore, the flux descriptor is expressed both by means of

attributes in the source or destination nodes, in the application or in the field of DiffServ that appears at the IP head. If the flow descriptor is specified by a source and destination IP address and the reach parameter is not specified, then we cannot know which is the entry point and the exit one that this traffic will use. In this case, the route can be calculated by means of routing policies previously defined. On the other hand, if the flow descriptor and the reach in the SLS are specified, then we would have the pairs IP source-IP destination and IP ingress-IP egress and the route that the IP packages must follow (IP source-IP ingress...IP egress-IP destination) would be clearly defined.

Traffic Envelop. It is a set of traffic conformance parameters describing how the packet stream should look like to get the guarantees indicated by bandwidth, delay and losses.
The envelop parameters or traffic adjustment are the base for the traffic conformity algorithm that is usually executed in the edge nodes of the DiffServ network. Some traffic adjustment parameters are: maximum rate, maximum and minimum transference unit, etc.

Excess Treatment. It describes how to treat the exceeding traffic, as for example, the traffic that is placed out of the conformity parameters. The exceeding traffic can be eliminated, delayed until the moment in which it covers the conformity parameters or be remarked with a particular value of the DSCP (yellow or red). In case there is not a specification about how to treat this kind of traffic, then it is eliminated in its default.

Performance Guarantees. They describe the service guarantee that the network offers to the packages of a flow descriptor considered particularly and in the topological/geographical limits that the scope parameter establishes. In this Thesis, four execution parameters are used; delay and jitter indicate the maximum packet transference. The packet loss indicates the loss probability that packages in conformity with the traffic parameters have from an initial point to a final one. Delay, jitter and losses are measured in respect of the traffic that accomplishes the traffic conformity. The bandwidth is a rate measured in destination.

Service Schedule. The start and end time when the network is available for the user, for example, when the service is available. The service plan is expressed as a collection of the following parameters: day hours rank, weekdays rank, year months rank, etc.

Reliability. It indicates the maximum number of times a year that a failure in the network can appear and the maximum necessary time to repair those failures.

Furthermore the seven Tequila parameters, we add an extra parameter called *Priority policy* that indicates if the policy server has some special policy or specific service to apply to a group of users. This field has priority over any other SLS field. Some examples of priority policies can be the offers that a company establishes, time frame, temporary prohibition in the use of a service, etc. Usually priority policies come from business goals or temporal failures.

The SLS parameters fix the value rank that a connection can use. This whole of eight parameters is called SLS profile. In this way, if the parameters that the user's connection requires are out of the values that the SLS profile establishes, then the network does not guarantee the connection.

The edge router also sends the parameters that the user's connection requires to the management platform. It also sends the required source and destination addresses. These addresses must be in conformity with the scope parameter belonging to the SLS profile that the user owns. The figure 3 sows an example about how the SLS profile parameters influence.

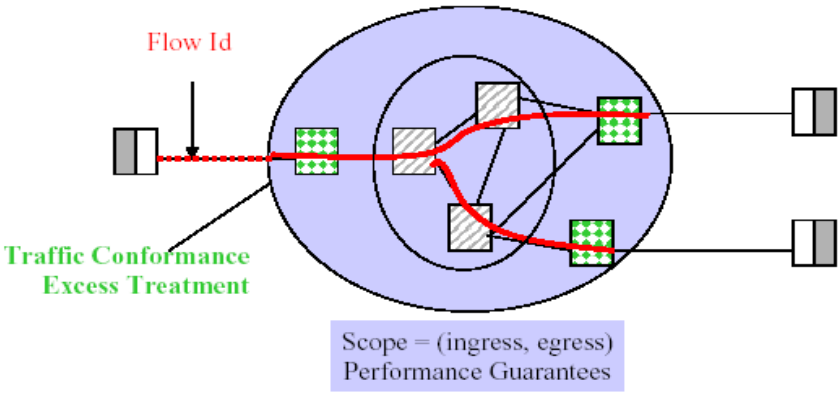

Figure 3. Scope and Performance guarantees of the SLS Profile.

3.7 Contribution in this chapter

The general scheme of the architecture used for a PBMS proposed system is presented in this chapter. These are the general contributions:

- Definition, design and implementation of a management system based on policies. Specification of requirements and network parameters for classes of services.
- Integration and conversion among network management policies and service management policies.
- Creation of SLS profiles.
- Specific contributions concerning the PBMS architecture elements:
    1. Use of LDAP directory
    2. Use of IIOP protocol from the CORBA scenario (of OMG) for the distribution of policies and parameters
    3. Use of intelligent agents to monitor the network. Implementation with JDMK development kit.

References

Libros

[Kilkki99] K.Kilkki. Differentiated Services for the Internet. Macmillan Technology Series. ISBN: 1-57870-132-5. 1999

[Verna02] D. Verna. *Policy-based Networking. Architecture and Algorithms*. New Riders. ISBN: 1-57870-226-7. November 2000

[Siegel00] J. Siegel. Quick Corba 3.Wiley Computer Publishing.ISBN 0-471-38935-8. 2000

Papers

[Anjali02] Tricha Anjali, Caterina Scoglio, et.al. *A New Path Selection Algorithm for MPLS Networks Based on Available Bandwith Estimation* QofIS 2002, Zurich, Switzerland, 2002.

[Barba02] Antonio Barba, Ernesto Sánchez. *An Architecture for Active Network Performance Management Based on Intelligent Agents.* Mobile Agents for Telecommunication Applications (MATA), Proceedings. Lecture Notes in Computer Science 2521 Springer 2002, ISBN 3-540-00021-6 Barcelona, Spain. 2002 pp. 94-104

[Brunner01] M. Brunner, J. Quittek, "MPLS Management using Policies" , Proceedings of the 7th IEEE/IFIP Symposium on Integrated Network Management (IM´01), May 14-18, Seattle, USA, 2001.

[Dulay01] Dulay N., Lupu E., A Policy Deployment Model for the Ponder Language. IEEE/FIP International Symposium on Integrated Network Management (IM' 2001). IEEE Press, Seatle, May 2001.

[Fritz99] J. Fritz Barnes, R.Pandey. "CacheL: Language Support for Customizable Caching Policies. Fourth International Web Caching Workshop (WCW99). California, USA. March 1999.

[Goderis00] Goderis D. et al. *Functional Architecture and Top Level Design.* TEQUILA Consortium Deliverable D1.1 September 2000. www.ist-tequila.org/deliverables.html

[Goderis00-1]Goderis, D., T'joens, Y., Zaccone, et.al. *Service Level Specification Semantics, Parameters and Negotiation Requirements* draft-tequila-diffserv-sls-00.txt, July 2000.

[Koch96] T. Koch, C. Krell, B. Krämer. *Policy Definition Language for Automated Management of Distributed Systems.* Second International Workshop on Systems Management. IEEE Computer Society. Toronto, Canada. June, 1996

[Li01] Y. Li, M. Chen, X. Jiang, L. Song. *A Logic-based Policy Definition Language for Network Management.* 12th International Workshop on Distributed Systems: Operations & Management (DSOM 2001) IEEE/IFIP. Nancy, France. October 2001

[Lobo99] Lobo J., Bathia R., Naqvi S. *A policy description Language*. Proceedings of AAAI Press. pp.291298. Orlando. USA. 1999

[Reyes00-1] Angélica Reyes, Antoni Barba *Gestión de encaminamiento basado en restricciones de Calidad de Servicio en Internet2* Union Radio-Scientifique Internacionale (URSI2000). Zaragoza España. 2000

[Reyes00-2] Angélica Reyes, Antoni Barba *Desarrollo de una aplicación de Gestión de encaminamiento en Internet2* Telecom2000. Madrid España. 2000

[Reyes 00-3] Angélica Reyes, Ernesto Sánchez, Antoni Barba *Routing Management application based in mobile agents on the Internet2* Eunice 2000. Holanda.2000

[Reyes01-1]Angélica Reyes, Antoni Barba. *Arquitectura de un sistema de gestión basado en políticas de servicio en Internet 2*. CITA2001. Cartagena de Indias, Colombia 2001

[Ribeiro99] C.Ribeiro, P.Guedes. SPL: An access control language for security policies with complex constraints. Technical Report RT/0001/99, INESC. Enero 1999.

[Sloman94] M. Sloman. *Policy Driven Management for Distributed Systems,* Plenum Press Journal of Network and Systems Management, vol 2, no. 4, Dec. 1994, pp. 333-360

[Trimintzios] Trimintzios P, et al. Architectural Framework for providing QoS in IP Differentiated Services Networks, 7th IFIP/IEEE International Symposium on Integrated Network Management.

[Virmani00] Virmani A., Lobo J., Kohli M.,*NETMON: Network Management for the SARAS softswitch*. IEEE/IFIP Network Operations and Management Symposium. April 2000.

Standards

[Hodges02] Hodges, J., and Morgan R., *Lightweight Directory Access Protocol (v3): Technical Specification*, IETF Request for Comment (RFC) 3377, September 2002.

[Rosen01] E. Rosen, A. Viswanathan, R. Callon. *Multiprotocol Label Switching Architecture.* IETF Request For Comments (RFC) 3031. January 2001

[Heinanen99] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski. *Assured Forwarding PHB Group*. IETF Request For Comments (RFC) 2597. June 1999.

[Moore01] Moore, E. Ellesson, J. Strassner, A. Westerinen. *Policy Core Information Model-- Version 1 Specification.* IETF Request for Comment (RFC) 3060. February 2001.

[Davie02] B. Davie, A.Charny, R. Bennet, K. Benson, et.al. *An Expedited Forwarding PHB (Per-Hop Behavior).* IETF Request for Comment (RFC) 3246. March 2002.

[Yavatkar00] R. Yavatkar, D. Pendarakis, R. Guerin. *A Framework for Policy-based Admission Control*. IETF Request for Comment (RFC) 2753. January 2000.

[Nichols01] Nichols K., "Definition of Differentiated Services Per Domain Behaviours and Rules for their Specification" RFC 3086, April 2001.

[IIOP] CORBA™/IIOP™ Specification:
http://www.omg.org/technology/documents/formal/corba_iiop.htm