A Policy Creation and Enforcement Environment for an IP Network

Mark McDonagh

M.Sc in Networks and Distributed Systems

Department of Computer Science

University of Dublin, Trinity College

Ireland

September 15, 2001

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: ____________________

Mark McDonagh B.A.

Date: September 15, 2001

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Signed: ____________________

Mark McDonagh B.A.

Date: September 15, 2001

Acknowledgments

I would like to thank my supervisor, Mr Vinny Wade for his guidance and assistance in the completion of this dissertation and to Mike Barry and all his team at Broadcom Eireann Ltd for their assistance.

Thanks to my family who have always supported me and to all my friends.

Abstract

There is an increasing number of applications designed to run on the Internet whose Quality of Service (QoS) requirement is higher than the current offering from the IP network that being best effort. There has been a call by many in the Internet Community to change the way in which the well trusted IP protocol works so as to provide a level of QoS better than best effort. Numerous protocols have been suggested that could enhance the QoS on the Internet. In changing the workings on the Internet we must carefully consider which is the best option. One approach to IP QoS is the use of policies to enforce QoS. One of the problems in using policies to support QoS is how to create policies on the basis of an Service Level Specification (SLS).

This dissertation describes the design and implementation of a framework which supports the creation and deployment of policies on to an IP network.

The framework consists of a number of components. The first component is a Policy Enforcement Point which is used to deploy policies on to routers. The second component is the Policy Management Tool, this component allows users to deploy a policy on to the network. The final component of the framework is the policy creation tool. The policy creation tool makes it possible for a user to enter a required SLS, the policy creation tool will suggest a policy which will meet the required SLS, or it will recommend a policy close to the required SLS and how to change it to match the required SLS.

The final part of this disseration describes an application that was developed which demonstrated the capabilities of this framework.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The Internet is operated by multiple Internet service providers (ISPs) whose responsibility is to provide the physical means by which data can be transported. An Application Service Provider (ASP) offers clients a service which is accessed over the Internet. The ASP is dependent on the ISP to transport their service. The relationship between the ISP and the ASP is crucial for the success of the ASP business model.

The ASP model of providing a product is becoming an increasing phenomenon for the distribution of software applications. When an application is provided as a service a client will need to have a guarantee in terms of the availability of this service. The guarantee that the ISP provides is know as the Service Level Agreement (SLA), the SLA is an agreement between the service provider and the client on the quality of the service provided by the service provider. The SLA covers all the statistics that the customer is interested in being guaranteed (the combination of these statistics should be the level of service that is required by the customer). The SLA can be divided into two parts, the first part of the SLA will be concerned with issues regarding the service provided itself i.e. will there always be a connection for the client. The second part of the SLA concerns itself with the Quality of Service (QoS) that the traffic generated between the ASP the and client experiences. In general when we refer to QoS it is referred to it in the context of IP traffic. QoS refers to a “set of capabilities that allow a service provider to prioritize traffic, control bandwidth, and network latency” [WSS$^+$01] QoS is one of the most important issues in Computer Networking. This is partly due to the ever-increasing number of applications being developed which are designed to run over a network as a service (e.g. Real Player, Voice Over Internet Protocol more commonly known as VOIP). A distributed computing framework for operation is a common feature that these applications share, however they require vastly different QoS levels from the network. These different QoS levels cause a problem for IP based networks, which are only designed to support one level of QoS that being “best effort”.

This chapter introduces the area of QoS and the need for QoS support in IP networks. The objectives of this dissertation are then stated followed by the technical approach taken to this dissertation.

1.2 Motivation

Policy Based Network Management (PBNM) is one of the approaches that has been developed to support IP QoS and will be the main protocol of investigation for this dissertation. PBNM is a new initiative in the area of network management, this technology is an emerging technology which promises much in terms of revolutionising the concept of network management. It is a rich vein for research into the practicalities of implementing such a system. This dissertation has been motivated by the issues involved in implementing a PBNM system and the authoring of policies which can provide QoS. If we consider the likely scenario in which a PBNM system will be deployed there will be a client of a network operator typically an ASP who will wish to have a SLA enforced between a source and destination, this SLA will take the form of a numerical specification which can easily be monitored (either the SLA is being met or not). To meet this SLA the network operator must be able to map the customers SLA to a policy that can meet the SLA. This is the core part of our research, how to match SLAs to policies.

1.3 Objectives

The primary goal of this dissertation is to investigate current approaches to QoS and particularly the use of Polices to support QoS and how QoS can be implemented across multiple domains.

The second objective of this dissertation to investigate the issues and challenges involved in the semi-automatic authoring of policies for different SLAs. These policies should be semi-automatically authored on the basis of a set of customer defined QoS parameters identified by the network operator. The final goal of this dissertation is to evaluate the application that is developed within this dissertation. The application will be evaluated under two headings, one, has it contributed to the state of the art in regard to QoS, including how our implementation of a PBNM system compares to the standard developed by the Internet Engineering Task Force, and two, has it provided a sufficient base for which other researchers can use the application as a basis for extending it to support the creation of more complex policies.

1.4 Technical Approach

The dissertation was undertaken with the following approach. First, the area of QoS was investigated including the areas of Integrated Services (IntServ), Differentiated Services (DiffServ), Multiprotocol Label Switching (MPLS) and the use of SLSs. Then second stage of research for this dissertation was the area of PBNM in association with IP Networks, also we investigated of the problems which are encountered when authoring policies. The third stage of our research was investigating how policies could be authored so as to reflect service level specifications, this stage also included investigating how these policies could be authored in a way so that they could be deployed on a router which is running the Alternative Queuing (ALTQ) routing software [Cho]. The fourth stage of the dissertation was the design of a framework that could generate policies which reflected the users required SLS and could be deployed on to a test network. The final stage of our research was the building of an application based on our design.

1.5 Overview of Dissertation

Chapter 1 *Introduction* This chapter presents the motivation for the research, the objectives of our research and finally the key technologies used in the dissertation.

Chapter 2 *State of the Art* This chapter provides an analysis of IntServ, DiffServ and Policy Based Network Management. The problems which are encountered in developing policies are also discussed. Finally the use of Service Level Specifications is discussed.

Chapter 3 *Design* This chapter discusses the requirements for the dissertation and details the architecture of the policy deployment system and the Policy Creation system. Different design strategies are also discussed.

Chapter 4 *Implementation* This chapter presents the implementation of the One Touch Management system, which includes the Policy Advisor and highlights its important characteristics.

Chapter 5 *Evaluation and Conclusion* This chapter evaluates the design and implementation of the One Touch Management system and the Policy Advisor. Further avenues of research in Policy Based Network Management are also discussed.

1.6 Summary

This chapter gave an overview of the need for improving the current type of QoS on IP networks, which is “best effort” to a mixture of “best effort” and guaranteed QoS. Though the mechanisms for creating QoS on IP networks have been developed, there is still a gap in the research in aligning the service level requirements of users with the mechanisms that support QoS. Finally the objectives of the dissertation were stated followed by the technical approach taken to the dissertation and then an outline of the dissertation was given.

Chapter 2

State of the art

2.1 Introduction

In this chapter, three different approaches to QoS are examined, IntServ, DiffServ and PBNM. PBNM is discussed in a more detailed fashion, as it is the protocol of investigation for this dissertation. In this chapter the technical problems which are encountered when authoring policies are explored and the concept of a SLS is also explained.

2.2 Quality of Service

2.2.1 What is Quality of Service

QoS refers to the ability of a network to provide differentiated treatment to selected network traffic over various technologies such as Frame relay, ATM, Ethernet, Sonnet and IP routed networks [RCD00]. In essence this means that the network will no longer be fair in its allocation of resources but will allocate resources to those that actually need the resource or to those who are willing to pay for it. For example QoS techniques could be used to provide a service where one could make VOIP phone calls with a QoS akin to a normal telephone call.

The use of IP QoS in the preceding usage is done in the context of applying the techniques of service engineering to a network so that it can provide a service in accordance with the goals of the network. At this early stage it is important to denote the difference of service engineering and resource engineering. Service engineering is concerned with engineering a system so that it can meet a level of service with some guarantee, whereas resource engineering is concerned with maximising the output of a system. In theory the two concepts will be used together, but in some cases the principles of resource engineering will be sacrificed for service engineering principles.

The formal way of specifying a QoS level between a client and a service provider is know as an SLA. An SLA is made up of two parts, a service level specification which will define the type of service the clients traffic will receive in terms of delay, jitter, throughput and loss. The second part of the SLA will be taken up by a contractual agreement which will describe the penalties that will be incurred if the SLA is broken. Normally what will happen is that a service provider will sign a contract with the client binding them to provide the type of service specified in the SLA and if this SLA is broken there will be a penalty clause enforced on the network operator.

2.2.2 The Need For Quality of Service

An application which uses the Internet for transporting data will normally operate better if its Internet connection is of high QoS, this means in theory that all Internet users will wish for a higher level of QoS. In practice the number of users who actually need a higher level of QoS is quite small. QoS enabled networks will only contain a small percentage QoS traffic and best effort traffic will continue to be the dominant form of traffic on IP network. The following are the reasons why there is a need to change the design of IP networks remembering that changing the design of the IP core network from a dumb network to a not so dumb one is a important change in design.

The first reason why QoS capabilities need to be to added to IP networks, is that regardless of how much extra bandwidth that will be added to the global network in the coming years there will still be problems in terms of QoS. The reason for this lies in the traffic usage patterns of IP networks, the difference in mean traffic and peak traffic in IP networks is large. This means that provisioning a network with resources which can deal with peak load traffic will mean a large degree of wasted resources during times when the network is operating at peak traffic load. The second disadvantage to scaling your network to meet peak traffic is the huge cost it would entail.

The second reason why IP networks needed to be enabled with QoS is that in recent years a new breed of Internet applications have been developed which rely on strict QoS requirements in order for them to operate efficiently, the current form of IP networks can not meet these requirements.

2.3 Quality of Service Architectures

Currently the Internet Engineering Task Force (IETF) is working on four ways of implementing QoS on IP networks. The first standard that has been developed is called IntServ.

2.3.1 Integrated Services

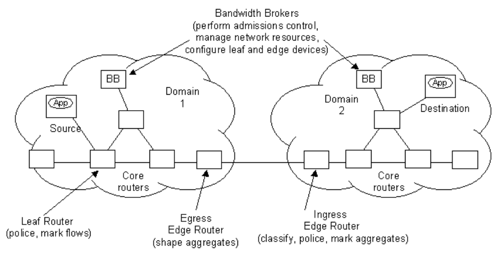
IntServ [Wro97] works by requiring applications to signal their service requirements to the network through a reservation request. IntServ uses Resource Reservation Protocol (RSVP) [BZB$^+$97] to reserve the required bandwidth throughout the network. This means the use of a simple queuing technique at the router which guarantees the specific traffic x amount of bandwidth. This standard has obvious limitations in that for example if one wished to set up a VOIP call from one side of the world to the other it would require a reservation to be set up for each router between you and the call recipient, when it is considered scaling this protocol up to thousands of users setting up reservations it would indicate that an extreme amount of state information would be stored in each router, which makes this protocol inappropriate over large networks.

For practical reasons IntServ in all likely hood will not supply us with a viable QoS standard. It has been suggested [Eur00] that IntServ is an appropriate QoS protocol for smaller networks i.e. corporate Intra networks which are connected to IP backbones. In our opinion IntServ is a very limited QoS protocol it is reasonable to suggest that it would be best implemented on a small network but all it can offer a network operator is the ability to reserve bandwidth, it lacks the abilities of PBNM which can limit traffic, block traffic, allow traffic only in times of low traffic usage. The other standard upon which the IETF is working on is known as MPLS.

2.3.2 Multiprotocol Label Switching

The Multiprotocol Label Switching (MPLS) [RVC01] is the second approach to QoS that is being developed by the IETF. MPLS operates on the principle

Figure 2.1: Differentiated Services Network [ST]



that routers are a bottleneck in the IP network, to reduce this bottleneck the MPLS approached the problem by seeking to reduce the operations done on the router. Each packet entering an MPLS network has an MPLS header attached to it, each router within the MPLS network then routes the packet according to the MPLS header. The MPLS header contains information which identifies which pre-determined path through the network a packet will take.

2.3.3 Differentiated Services

The method in which Differentiated Services (DiffServ) [BBC$^+$98] provides QoS is by separating the IP packets into different classes known as Per Hop Aggregates (PHA), then configuring all network devices to give a specified QoS for that class of packets. Fig 2.1 is a representation of a typical DiffServ network. A DiffServ domain consists of a set of DiffServ Nodes which have a common set of Per Hop Behavior (PHB) implemented on each Node. PHB is the technical term for how packets of the same PHA are treated at the routers. Next is an overview of what each element of the DiffServ does.

DiffServ Ingress Nodes

All traffic entering a DiffServ domain must enter through a DiffServ Ingress Node (Entry Node), the DiffServ Ingress Node has multiple functions [BBC+98]. The first function that is carried out at the DiffServ Ingress Node is traffic classification. This involves separating the incoming traffic into different PHA based on the packets profile; each PHA has its own unique code point associated with it. This code point is marked in the Type of Service field in an IP packet. The second function that is carried out on the DiffServ Ingress Node is Traffic conditioning which involves shaping the traffic. Traffic shaping involves two techniques, firstly the fragmentation of larger packets into smaller packets and secondly traffic metering. The purpose of traffic metering is as follows: traffic entering a network often enters in a bursty form, when traffic enters the network in this manner it is said to enter in trains, the purpose of metering is to space out the trains prior to transmission by temporarily storing packets in buffers to make sure the network is not overloaded. The difference in the QoS experienced by the different classes of traffic is due to the queuing algorithms, which are enforced on the routers. There are a variety of techniques used to enforce QoS on the routers. The design of DiffServ was very much influenced by the design principle which influenced the design of the IP network itself that is to keep the core network as simple as possible and keep as much of the complexity as possible to the edges of the network, hence this is where the largest degree of complexity lies in the DiffServ network.

DiffServ Core Nodes

The purpose of the Core Nodes is simply to forward the packets according to their PHA [BBC+98]. The PHA will be forwarded on the router according to a matching PHB. The PHB will include some type of queuing algorithm such as Class Based Queuing or FIFO, these queuing algorithms will be explained later on in this chapter.

DiffServ Egress Node

The purpose of the DiffServ Egress Node (Exit Nodes) is to perform traffic conditioning functions on traffic forwarded to a connecting DiffServ domain [BBC+98]. This traffic conditioning is needed because if there is traffic going from one DiffServ Domain to another DiffServ domain then there needs to be an agreement as to the amount of bandwidth the neighbouring DiffServ Domain will accept of each PHA. To avoid this agreement being broken there needs to be a traffic conditioning function at the Egress Node so as to prevent excess traffic entering the neigbouring DiffServ Domain.

Bandwidth Broker

The Bandwidth broker [BBC+98] is a component in the DiffServ network which supports DiffServ across multiple DiffServ Domains. The most apparent means of creating a QoS class that travels between multiple DiffServ domains would be to form a multilateral agreement between all the domains that the traffic flows through. This approach was regarded as infeasible due to the belief that multilateral agreements rarely work [BBC+98], instead it was suggested that a bilateral agreement between neighbouring DiffServ domains between the source and destination would be a more effective approach. The idea of the Bandwidth Broker is as follows, each DiffServ domain has a Bandwidth Broker associated with it. The Bandwidth broker is responsible for configuring the Ingress Nodes, when a DiffServ domain requires a class of QoS across multiple domains it sends a request to its neighbouring DiffServ

Domains Bandwidth Broker with the type of service required, the Bandwidth Broker then checks has it the bandwidth necessary to fulfill the type of service required without damaging other QoS classes. If it does, it configures its Ingress Nodes so as to allow the required bandwidth. It is important to note that the Bandwidth Broker must have a knowledge of traffic engineering to successfully do the job.

Inter Domain DiffServ

The support of QoS service across multiple DiffServ domains is a key requirement for the successful implementation of DiffServ as a QoS solution. The main issue is how to accommodate a user who has an SLS that needs to be enforced across multiple domains.

The first thing to remember is that DiffServ is only designed for four to five quality of service classes. In the DiffServ model what will happen is the request for a QoS level will be signaled to its neighbouring DiffServ Domain bandwidth broker and a peer to peer agreement will be formed to build the level of QoS required. Note because of the limited PHBs, it is unlikely that a user will be able to get their required SLS.

The evolution of Inter Domain DiffServ as outlined has some flaws, first of all is it realistic to expect that network operators will allow a system to dynamically auto configure there network. Another question which needs to be solved is how to design a system that can guarantee that by adding more traffic to the network other SLSs will not be broken. A more realistic view of the evolution of inter Domain DiffServ would be a scenario where each DiffServ Domain would have an agreement with its neighbouring DiffServ Domains which would see a limited number of QoS classes being negotiated between neighbouring DiffServ Domains. These QoS classes would be fixed in terms of there SLS, then when a network operator receives a request for an SLS which crosses multiple DiffServ Domains they would assign the traffic to one of the predefined QoS classes. If this approach is adapted to DiffServ across multiple Domains, reconfiguration only occurs at the source DiffServ Domain. Another question which arises with Inter Domain DiffServ is what happens when traffic is passing through multiple DiffServ domains, the first step of the problem is outlined above where there is an agreement with our neighbouring DiffServ Domain is reached but what happens to our traffic when it leaves our neighbouring DiffServ Domain. If the assumption is made that our neighbouring DiffServ domain has agreements with its neighbouring DiffServ Domains and so on until we have a picture where there are agreements in place between the DiffServ Domains between source and Destination. This is the required scenario we will need if strict SLS are been guaranteed. This scenario is a subject for more research in the area of traffic engineering.

In Review

DiffServ is a more practical QoS implementation than IntServ, it is a simple solution, which avoids the complexity that has beset IntServ. One shortcoming of this standard is that the number of PHB is limited. Another limitation of DiffServ is the fact that core routers are kept as simple as possible though this is seen as one of the main benefits of the design of the IP network, it is a hindrance for DiffServ due to the fact that traffic is not routed according to a predefined pattern hence the SLS of a PHB is the worst SLS that the traffic could take through the network. The disadvantages that were outlined in regards to DiffServ are solved with the Policy approach to network management.

2.4 Policy Based Network Management

2.4.1 What is Policy Based Network Management

> Policies are plans of an organisation to achieve its objectives. A policy is a persistent specification of an objective to be achieved or a set of actions to be performed in the future or as an on-going regular activity [Slo].

The use of policies for management is an approach to enterprise management which aims to reduce the task of managing systems which are growing in size and complexity.

Policy based management has been applied to security management as well as network management. In this dissertation we will discuss the use of policy base management in regard to network management. PBNM is a hybrid version of the DiffServ Model, instead of a set of predefined PHBs which have traffic types mapped to PHBs, with PBNM the PHB are not predefined and are normally configured by the network operator, PBNM offers more freedom in the generation of policies than DiffServ.

A policy is defined as a combination of one or more sets of rules containing conditions, which dictate actions to help automate network, system, service, and process management [Com01]. As mentioned above a rule consists of a condition, which has a corresponding action, a condition allows an administrator to specify a constraint on a rule. A condition will normally take the form of an IP address, IP port (both source and destination), date and time, application traffic etc. An action specifies how packets that meet a corresponding condition are treated. PBNM allows network managers to create policies, to define how resources or services on the network can or cannot be used. At this stage it is important to clarify the difference between an order and a policy. An order could be given to a person to boil the kettle. This would be deemed an order whereas a policy would be defined as to boil the kettle every day at one o' clock. In this scenario the condition is met when the time is one o' clock and the action is that the kettle is boiled, this policy will happen continuously as long as the policy is deployed whereas the command only occurs once.

2.4.2 Policy Architecture

The IETF Policy working group [SWM+99] has defined a policy schema for the enforcement of a Policy Based Network Management System, which consists of four parts:

- A Policy Management Tool
- A Policy Repository
- A Policy Consumer
- A Policy Target

The Policy Management Tool

The Policy Management Tool is divided into six parts. The first part of the Policy Management Tool is the Policy Editor, which is used for entering, viewing and authoring policies. The second part of the Policy Management Tool is the Policy Translator, which is used for the translation of general policy specifications into policy rules that the Policy Consumer can use. This translation will not provide a translation on which the Policy Consumer can directly act on, as it will be device and vendor independent but it will be left to the Policy Consumer to translate this interpretation into a form, which can be directly enforced on the Policy Target. The third part of the Policy Management Tool consists of a function which can validate a policy, this validation will validate the policy to prevent errors in two regards firstly it checks that the data types within the policy have been correctly used (i.e. an IP address which is not formed correctly), the second validation is the checking of the policy in regards to semantics. This will ensure issues such as the source and destination address do not have the same IP address. The fourth function of the Policy Management Tool is a Global detection function, which is used to see whether or not a newly added policy conflicts with other policies, policy conflicts are detected on the basis if resources have been fully allocated within the installed policies and the system tries to install a new policy which tries to allocate some resources in a different way, there will arise a policy conflict.

The Policy Repository

The second part of the policy schema is the Policy Repository, this part of the system is used to store policies once they have been translated and verified. The Policy Repository should be available for Policy Consumers to download policies.

The Policy Consumer

The third part of the policy schema is the Policy Consumer, the purpose of the Policy Consumer is to acquire policies from the Policy Repository, deploy the policies to policy targets and sometimes the translation of these policies into a format which is usable by a policy target. The final part of the policy schema is the policy target.

The Policy Target

The Policy Target is the packet forwarding device itself, the Policy Target should operate as specified by the Policy rule that is being applied to that Policy Target.

2.4.3 Benefits of Policy Based Management

PBNM promises many benefits to those that use it to manage their network, in this section we analyse the benefits that promoters of PBNM claim can be derived from such a system and analyse how realistic these claims are.

> PBNM helps you translate business objectives into network support by
>
> - capturing these business level objectives as policies, and then distributing them into the managed network environment.
> - using consistent policies to control heterogeneous devices from diverse vendors.
>
> [Com01]

The first claim is currently unrealistic at this stage of the evolution of PBNM systems if you are capturing business objectives in terms of numerical figures. Currently PBNM systems do not allow the mapping of business objectives in terms of numerical figures straight into policies that can meet these business objectives. If we define business objectives to mean that the network should operate "efficiently", for example that Telnet traffic would always get precedence over HTTP traffic then we can say that PBNM does enable the translation of business objectives on to the network.

The second claim is that PBNM allows you to use consistent policies to control heterogeneous devices from diverse vendors. This is a point which is discussed further in section 2.5.7 but the fact that there is no standard policy language currently makes it difficult to claim this benefit.

> PBNM minimises administrator effort by
>
> - managing your network at a higher level than was previously possible
> - eliminating the need to learn the QoS configuration requirements of each individual device from every vendor through the use of general policies
> - abstracting the details of how a resource provides prioritised service
> - helping you gain better control of the network
>
> [Com01]

The first claim is true in that PBNM configures policies with a group of routers in mind and not at the router level as was done in the past. This is a benefit because we begin to think of the global effect of a “change in configuration” and also we can enforce a change in configuration on to the whole network by enforcing just one policy.

The second and third claim is true if the PBNM system you employ supports a single policy language which can operate over heterogeneous routers. This is an enormous benefit to a network administrator because

- you can use heterogeneous network devices in your network.
- it prevents you from having to learn the complexities of a variety of policy languages.

The third claim is also true in that with a PBNM you can gain a level of control over your network that was not possible in the past. Without a PBNM system your network operated according to the underlying network protocol, but with a PBNM system your system operates according to the underlying network protocol with allowances made for the way you want your network to operate.

PBNM helps keep network costs down by

- better utilising existing network bandwidth
- eliminating the need for over-provisioning the network
- delaying investments by extending the life of your current infrastructure

[Com01]

The first claim that a PBNM system will better utilise network bandwidth is true is that a PBNM system will help utilise a network more efficiently because we can restrict traffic that might be regarded as “discretionary” from taking bandwidth from traffic that may be regarded as “critical”. In the past when the user of a network was unhappy with the level of QoS they were experiencing at peak loads (which may be occurring during 1 % of the time) the network operator would have two choices

- reduce the number of network users
- increase the capacity of the network, with a PBNM system it is possible to reassign resources in the network to meet the required QoS requirement without having to add network resources which will only be used during peak periods.

The final claim in regards to PBNM is also true in that with a PBNM system you can delay investment in your network. This is due the fact that peak traffic in an IP network varies greatly from mean traffic, without a PBNM system in the past if your peak traffic could not be handled by your current network you would have to upgrade your network, now with a PBNM when problems occur due to capacity at the peak rate instead of upgrading your network you can develop a policy to prioritise your traffic to overcome the problem and wait to upgrade your network until mean traffic is closer to the network capacity.

2.4.4 Disadvantages of PBNM

PBNM also has disadvantages. Caruso [Car99] sites two main problems with PBNM firstly, because there is no standard policy language to implement policies consistently on all routers, it would require a user to have all his devices from the same vendor which Caruso believes would be difficult in an environment where most networks are made up of devices from different vendors. This is a valid point which will be discussed in more detail in section 2.5.7. The second difficulty that Caruso states with PBNM is in the case where it is implemented on a corporate network, Caruso foresees that a company will have difficulties in deciding who deserves a higher level of QoS. This is essentially a problem of politics, in our opinion this problem will not be a set back for the development of PBNM systems but will be overcome when the systems start getting deployed.

2.4.5 PBNM Systems Implementations

In this section we will offer an overview of the PBNM systems that are available from network vendors.

Cisco QoS Policy Manager 1.1

Cisco's contribution to the PBNM world comes in the form of a product called QoS Policy Manager [Inca].

Policy Creation

Cisco provides two techniques of creating QoS policies. The first technique of creating a policy is through using the Modular QoS Command Line Interface (CLI) [Incb], as the name suggests it is a syntax which can be used to define a policy on the router itself. The main disadvantage of creating policies in this way is that each router has to be configured independently, another disadvantage of this technique of policy creation is that the user must learn the propriety configuration syntax before he can create the commands. The second way in which policies can be created for Cisco devices is through using the Graphical User Interface (GUI) policy creation environment that comes with the Cisco QoS Manager. The GUI enable users to create policies which the system translates into the Modular QoS CLI. Here are the following steps which are taken when creating a Cisco QoS policy.

1. Select the devices that the policy is to be deployed on.
2. Change the QoS property on each router to the queuing algorithm required.
3. Create a filter which will dictate the traffic which the policy is to effect.
4. Select the parameters for the queuing algorithms which will effect each host associate with the policy.
5. Save the Policy.

HP OpenView PolicyXpert

The HP OpenView Policy [Pac] is Hewlett Packard's contribution to the PBNM market. The PolicyXpert architecture has five major components in its architecture. The first component is the policy console which is used to create and manage policies, the second component is the policy server which stores policies authored in the console and stores status information provided by agents associated with every policy server is a Policy Decision Point (PDP). The PDP evaluates policies, their rules, and their targets in order to render policy decisions to any agent that is requesting that the server make the decision. Attached to every policy server is a policy database for storing policies, this database is a proprietary-format database. The final component of the PolicyXpert architecture is the policy agent, policy agents are used to deploy policies to their target resources.

- Communication between the policy console, the policy server, and the agent is in the form of the Common Open Policy Service (COPS) [DDCH+00].

- The PolicyXpert architecture is vendor independent so that it can manage all types of routers etc.

Policy Creation

HP offers a policy environment quite similar to that offered by Cisco, it offers a propriety command line interface and also a GUI for policy creation. A policy can be created through the GUI using the following steps.

1. Select the queuing algorithm that is to used in the policy.

2. Enter the name of the policy.

3. Set the parameters for the queuing algorithm.
4. Select the traffic profile that will be effected by the policy.
5. Specify the time the policy will be effective for.
6. Save the Policy.

An important feature of HP PolicyXpert is that it does not allow more than one policy to be defined on any single router network interface card.

The policy creation environment in Cisco's QPM and HPs PolicyXpert are similar in style, both offer a CLI syntax and a GUI for policy creation. One major disadvantage that is associated with both implementations is that they fail to allow users state their objectives in the process of policy creation. Both lack the ability to accept the users service requirements and to create a policy which is based on the users requirements.

2.5 Authoring Policies

The main objective of this dissertation was to research how we could generate policies that could reflect a required SLS. For this research we needed to investigate the process how we author policies and what we must consider when we author policies. Moffett and Sloman [MS93] classify policies into two types, imperatival policies and authority policies. Imperatival are policies which cause actions to be initiated, and authority policies are policies that cause actions to be given authority to be carried out. For example an imperatival policy would be a policy which would cause packets to be dropped from a router whilst an authority policy would give a network administrator the authority to configure routers. Both imperatival and authority policies have five types of attributes which are as follows:

- Modality
- Subject
- Target
- Goal
- Constraints

2.5.1 Modality

A policy can have one of the following modalities:

- positive authority (permitting) e.g. allowing a user to transmit HTTP traffic.
- negative authority (forbidding) e.g. to forbid a user from transmitting HTTP traffic.
- positive imperatival e.g. requiring or obliging that an agent must do something.
- negative modality i.e. deterring an agent from some action.

2.5.2 Subject

The second type of attribute a policy can have is a subject attribute which defines a set of users, these users are the set of users who have the imperative or authority to carry out the policy goals within the limits defined by the policy constraints. e.g. the user whose IP traffic is effected by the policy.

2.5.3 Target

The third type of attribute a policy can have is a target objects attribute, which defines the set of objects at which the policy is directed. e.g. the set of routers that a policy is directed at.

2.5.4 Goals

The final policy attribute is the goals attribute which specifies what the policy should achieve in abstract terms but does not identify how to achieve these goals. If we relate this to IP policies this would represent the SLS that a policy meets.

2.5.5 Constraints

The constraints of a policy will define where and when a policy can be applied. For the policies we will create the constraints will be to what router the policy will be applied to and at what time period of our policy will be applied.

2.5.6 Policy Conflicts

Moffett and Sloman are the leading researchers in the area of policies, they have been researching the area of using policies to automate the management distributed systems. Their work is of importance because it examines the problems incurred when using policies for management. In this section we examine their work and how it can be applied to the use of policies in the management of IP networks.

When there is more than one policy in a system the possibility arises that two policies will conflict. Conflict is a key issue in the creation of policies this is because we may know what the effect of enforcing a single policy will be, but when the effects of multiple policies being installed at the same time are considered it complicates the matter.

Moffett et al regard overlap as the key to understanding conflicts. Overlap is said to occur when the subjects and target objects belonging to two policies overlap. There are four types of overlap. The first type is called double overlap which occurs when both the subjects and the target objects of another policy overlap, the second and third type of policies are called Subject overlap and Target overlap and the final type of overlap is called Subjects-Targets overlap where the subjects of one policy and the target object of another policy overlap.

The following are the types of policy conflicts which can occur.

Positive-Negative Conflict of Modalities

This occurs when there is a triple overlap of target objects, subjects and goals, i.e. when a subject is both authorised and forbidden for the same goal on an object. For example if a policy is created that allows a user to transmit HTTP traffic whilst at the same time there was a policy which forbid a user from transmitting HTTP traffic.

Conflict between Imperatival and Authority Policies

This type of conflict also requires a triple overlap of target policies, subjects and goals. It occurs when a subject is both required to initiate and forbidden to carry out an action.

Conflict of Priorities for Resources

This type of conflict occurs when there is a conflict of requirements for a limited or single resource whenever concurrent access is attempted. This is a conflict of actions not policies. This situation could occur when we entitle each user to conduct a VOIP call on the network, but if a situation arises where every user wishes to conduct a VOIP call at the same time a conflict of priorities for resources occur.

Conflict of Duties

This type of conflict will occur when two positive authority policies are in the double overlap relationship in which both the subjects and the target objects of the two policies overlap, there is a possibility that a subject can perform two operations which are defined by the application as conflicting, upon an object. Sloman et al describe a conflict like this occurring where an application defines that the person who creates a cheque cannot be signed by the same user, and two policies are created which entitle the same user to create a cheque and sign a policy.

Conflict of Interests

When the subjects of two authority policies overlap, this implies that the same subject can perform management tasks on two different sets of targets. In some application-defined circumstances this will lead to a conflict of interests. Moffett and Sloman [MS93] describe this scenario with the example of a merchant banker who is advising one client on a takeover bid whilst advising another client on an investment decision which would influence the take over bid.

Multiple Managers

When the target objects of two policies overlap there is a potential conflict arising from multiple managers of a single object, when the goals of the policies are semantically incompatible. In our opinion this type of policy conflict will be the type of policy which is of most concern when creating policies for IP networks. This is the type of conflict which occurs when two separate policies are deployed on to a single network and the goals of both policies cannot be met on this network.

Self-Management

This type of conflict occurs when the subjects of one policy overlap with the target objects of another policy. This situation would occur where the network administrator has responsibility for creating policies for himself, i.e. the network administrator is the subject of a positive authority policy which grants it permission to create policies for subjects, in this situation the network administrator is also the target object of a policy.

2.5.7 Policy Language

The language which is used for creating policies is of great importance. Under ideal situations there would be a standard policy language for authoring policies, this would enable the scripting of just one policy for a network which could then be deployed on to all routers regardless of router vendor. Unfortunately this is not the case, most of the large networking vendors have there own propriety language for the creation of policies. Not having a standard policy language means that network vendors are free to implement there own proprietary queuing algorithms, this has the following consequence if a user wishes to use a proprietary queuing algorithm in a heterogeneous network they will only be able to deploy the policy on to the routers which support the proprietary queuing algorithm. Service Activator from Orchestream, Inc [Incc] overcomes the problem of a lack of standard policy language by adding another level of indirection to their policy creation process. Orchestream have created their own policy language which is used for creating policies, when policies are deployed they are mapped to the proprietary policy language of the device the policy is to be deployed to.

2.6 Role Based Network Management

Emil Lupu [Lup98] of the Imperial College, London has developed the concept of a Role-Based Framework for Distributed Systems Management, at the centre of his framework is the use of policies to build roles. A scenario is envisaged where a role would be defined as a collection of policies, where the bearer of the role would be entitled to the service specifications of the policies attached to their role. This use of policies would seem to be feasible when they are used for security purposes. The main source of policy conflicts when roles are used for security purposes are Conflict of Interests whereas when a Role Based approach is used for network management the conflict problem is one of Conflict of Resources a type which cannot be easily solved without degrading the quality of service attached to some roles. If we were to use a Role Based Framework for managing our network we could use it for two scenarios.

Scenario A

Roles would consist of policies that would restrict a users network usage, for example a role would have associated with itself a set of policies that would for example limit the holder of the role to x or less amount of HTTP bandwidth, x or less amount of Real Player. In this scenario we have a role which is made up of policies which have a policy Modality of negative imperatival.

Scenario B

In this scenario a role would consist of policies which would offer guaranteed SLSs to holders of roles. For example a user would be guaranteed x amount of HTTP bandwidth at all times, or a role user would be entitled to make a VOIP call at any time with a guaranteed level of QoS. These types of policies are known as positive imperatival policies.

The first scenario in our opinion is a more realistic application of role based to network management; in the first scenario we are not committing any network resources unlike the second scenario where we are offering a level of service. The crux of the problem with the second scenario is that we would have to have resources reserved to satisfy the SLS of all the role users constantly this would have deteriorated the value we could gain from our network. The problem is best denoted by an example, it could be possible that a network operator would receive a request from a user for a IP video conference to be set up within the network with a certain SLS, though it might be within the bounds of the network to provide this SLS the network operator may have to refuse this request on the grounds that if this request was accepted it may mean the SLSs which are associated with users roles could be broken, if all the users exercised their entitlements that are associated with there role.

2.7 Service Level Specifications

For the successful implementation of QoS in a Multi-domain market there will need to be a standard set of semantics for defining the level of QoS required by users. A SLS is the term used to represent the parameters which define the level of QoS experienced by a flow of traffic. The need to formalise the set of parameters which define a level of QoS has been recognised by many within the Internet world, to this end a Birds of a Feather (BOF) has been set up with the view to convene a Working Group [RCD00] so as to agree upon a set of standard semantics. The reason we need to have a standard set of semantics was justified as being two fold firstly a standard set of semantics would enable the automation of the service negotiation process and secondly the design and the deployment of services across a multi-vendor and multi-provider environment requires a standardised set of semantics for SLSs being negotiated at different locations

- between the customer and the service provider
- within an administrative domain (for inter-domain SLS negation purpose)
- between administrative domains

Obviously the formal specification of SLS will be an important asset to the success of this dissertation. The SLS BOF have developed a structure for an SLS with the following four headings.

2.7.1 The Common Unit

Each SLS will have a single Common Unit, the Common Unit provides information which can identify the provider of the SLS and the receiver of the SLS, it also contains information as to the time period the SLS is applicable for.

2.7.2 The Topology Unit

The Topology Unit is broken down into two parts, which are used to describe which points in the network this SLS is available from.

SAP Sub-Unit

A Topology Unit will have a single SAP unit, a SAP unit will contain a list of end points for traffic using this SLS, these end points could be a source or destination point. Because of the complexity which may be involved in defining an end point in the list of end points each end point is assigned a serial number which can be used throughout the rest of the SLS. This avoids the complexity of describing fully an end point every time it is used.

Graph Sub-unit

A Topology Unit may have multiple Graph Sub-units, a Graph Sub-unit describes the source and destination of the traffic. Typically we would consider that an SLS would be between a single source and destination this would be the simplest type of Graph Sub Unit, consider an ASP with multiple clients instead of having a different SLS for each of its clients and having to have each of these SLSs monitored, one SLS could be used for all the clients hence reducing the number of SLSs to be monitored to one. This makes it possible to describe a source with multiple destinations.

2.7.3 The QoS Unit

The QoS Unit is used to define actual QoS that the traffic experiences. It is broken down into four parts as follows

Scope

The scope describes which Topology Unit that this QoS Unit is related to.

Traffic Descriptor

The Traffic Descriptor is used to describe the traffic that this SLS is to be applied to, normally it will take the form of the DiffServ code point.

Load Descriptor

The Load Descriptor is used to express the bandwidth that can avail of this SLS, and how the excess bandwidth is to be treated.

QoS Parameters

The QoS parameters are the core of the SLS, this is where we express our requirements in terms of Bandwidth, Delay, Jitter and Loss.

We view the development of a standard set of semantics for describing QoS as a positive step in the evolution of QoS. It is important that there would be universal acceptance of this standard, this would mean that a customer could easily compare the cost of the same SLS between different network operators. Another important issue that requires further research is how to measure the SLS, questions that need to be answered are, will the traffic be monitored constantly or periodically, and in what detail will we measure QoS parameters.

The final question that arises in the area of SLSs is how does a network operator guarantee an SLS. If we consider the scenario where a customer comes to a network operator with an SLS to be met, the network operator will find a policy that could meet this SLS. In this situation a network operator has a few strategies, firstly the network operator could rely on mathematics to prove that an SLS will be met, at its simplest this would involve if x bandwidth is required in the SLS and x+y is reserved then this policy should be satisfactory to this SLS. Another strategy would be to benchmark the policy, this would involve measuring the QoS that the policy is experiencing and then offering that level to the customer as the SLS. This strategy has the obvious problem in how do you fully benchmark a policy so as to be sure that the SLS will be met in all circumstances, to do this would require flooding the network with a lot of "dummy" traffic a process which is most circumstances would be impossible in a production network. The last strategy would be to do some benchmarking on the policy and then offer the customer a statistical guarantee that the SLS will be met with a certain statistical guarantee.

2.8 QoS Mechanisms

In this section we discuss two different types of algorithms which are used for the provisioning of QoS.

2.8.1 Queuing

The first types of algorithms we will look at are queuing algorithms, these are algorithms which define how packets that are in a router are queued whilst waiting to exit the router.

First In, First Out Queuing (FIFO)

This queuing algorithm is the simplest of all the queuing algorithms we will consider. Using FIFO packets are stored when the network is congested and forwarded in order of arrival when the network is no longer congested. FIFO has two main disadvantages, firstly it makes no decision about packet priority and secondly it provides no protection against ill-behaved applications. For example if a router was operating under FIFO and only two flows of traffic were using it, one flow a UDP flow and the other flow a TCP flow it is possible that the UDP would get the majority of the bandwidth even though the TCP traffic is of higher priority.

Priority Queuing (PQ)

PQ operates by dividing the traffic into four queues based on their premarked priority, which could be High, Medium, Normal or Low based on an assigned priority. The algorithm gives the higher priority queues absolute preferential treatment over low-priority queues. With this type of queuing it is possible that the lower queues might be completely starved unless there is a restriction place on the amount of high priority traffic allowed into the network. This queuing algorithm is useful because the longest traffic of high priority has to wait before transmission is the length of time for a packet to be sent.

Custom Queuing (CQ)

CQ allows you to divide up your traffic into up to 16 different queues, each queue is given a percentage of the link bandwidth each queue is then serviced in a round robin fashion according to their percentage. This algorithm is useful because we can guarantee the amount of bandwidth a flow will get.

Class Based Queuing (CBQ)

CBQ allows the partitioning and sharing of a link, it does by creating classes of traffic in a hierarchical fashion on the router. An advantage of CBQ is that you can assign an amount of bandwidth to a traffic profile and it will be guaranteed that the amount of bandwidth but also it can use any bandwidth that is not being used on the link.

Weighted Fair Queuing (WFQ)

WFQ operates by assigning an independent queue to each flow of traffic. Each queue is served in a round robin fashion but queues which are deemed to be of higher priority are given more of the bandwidth. The thinking behind WFQ is that high priority queues will not require too much bandwidth.

2.8.2 Congestion Avoidance

In a QoS architecture we can only offer a level of QoS for a limited amount of traffic, so as to limit this amount of traffic we must have a way to control the amount of traffic in our network we do this by using Congestion Avoidance algorithms.

Random Early Detection (RED)

RED [FJ93] is used to prevent queues from becoming too large on routers, RED works by monitoring the queue size on a router and when the queue reaches a certain size it randomly discards some packets. RED reduces the delay on a router because queues are not allowed too get to long, it also improves the efficiency of a router because without the RED algorithm the router would wait until the router is full and any packets which then arrive will be dropped, in this scenario it is quite possible that all the TCP connections will have some of their packets dropped and will then reduce there transmission using the Slow start algorithm. This has two implications firstly it will take time for the TCP connections to increase their transmission so as to effectively use the bandwidth of the router and the second implication is that it is likely that this pattern of all the connections reducing their bandwidth at the same time will occur because a TCP connection will continuously increase its bandwidth until it detects congestion.

Token Bucket Regulator

The token bucket regulator operates by creating a token every time period, for a packet to be transmitted there must be a token for it to be sent, tokens can be accumulated in a virtual bucket. The capacity of this bucket dictates the largest possible traffic burst, used appropriately it can be used to reduce delay given the fact the longest delay a packet being dropped is the period of time it takes to empty a queue.

2.9 Summary

In this chapter we covered the different approaches which may be taken to implementing QoS on an IP network. PBNM was examined in more detail than the other QoS protocols. We discussed the different problems which can occur when creating policies, also we looked at how policies can be used to define the capabilities of a role. Finally we examined how the concept of an SLS, where SLS is used to record a level of QoS.

Chapter 3

Design

3.1 Introduction

In this chapter we discuss the requirements for the policy deployment and creation environment that we are going to design. In the second part of the chapter the design of the two components is discussed.

3.2 Requirements

3.2.1 Motivations Revisited

Before the requirements for this project are discussed in-depth it is worth revisiting the motivations and objectives of this dissertation. The primary motivations for this dissertation was the emergence of different QoS protocols and how they could be combined in a overall QoS architecture. The second motivation for this dissertation is the need to reduce the task of authoring QoS policies whilst at the same time how to create policies which reflect SLS requirements.

Objectives

The Primary objectives are:

- The first objective of this dissertation is to investigate the different approaches to QoS, in particular the PBNM approach to QoS.
- The second objective of this dissertation is to investigate the issues and challenges involved in the semi automatic authoring of policies to support QoS, part of this objective would be to develop an application which can demonstrate the results of investigation.
- The third objective of this dissertation is to evaluate the application that was developed for this dissertation.

Based on the above objectives we can divide up the requirements for this project into two main sections, the first section being the requirements for the policy deployment system and the second section being the requirements for the policy creation environment.

3.2.2 Requirements for Policy Deployment System

In this section we will explore the requirements for the policy deployment system.There were no non-functional requirements for the policy deployment system.

Functional Requirements

- The system should allow a user to enter the configuration of the `PEP` from a GUI.
- The system should allow a user to deploy a policy by entering the name of a policy and pressing a deploy button.

- The system must be able to store policies in a persistent format, the policies must be accessible by Policy Enforcement Points.
- The system should be able to change a routers configuration so as to reflect a change in configuration.

3.2.3 Requirements for the Policy Creation Environment

In this section we state the requirements for the policy creation environment.

Functional Requirements

- The system should suggest a policy to a user on the basis of an SLS.
- The system should be able to show a user the parameters of a particular policy that has been suggested.
- The system should allow a user to save a policy to the policy repository.
- The system must create its policy in a form which may be deployed on to an ALTQ router.

Non Functional Requirements

- The system should allow a user to view the policy which will be installed on a particular router.

3.2.4 Constraints

The system was developed as a test-bed for our policy based network management system, its typical use will be on a test bed network in a trusted environment. This means that we excluded security considerations from the requirements of the project. For example if we were to deploy this system on to a production network we would need to consider the following

- User authentication for users configuring Policy Enforcement Points.
- User authentication for users adding policies to the Policy Repository.
- User authentication for users deploying policies.
- Secure means of changing the configuration of a router.
- Passwords for routers which are stored in the system would need to be stored in a secure manner.

3.3 The Design of a PBNM System

This section of the chapter is concerned with the design of our framework, it is divided in to two sections, the first section describes how a PBMM system may be developed to provide a framework where we can use policies to manage our network and the following section describes a design for creating the actual policies on the basis of an SLS that will be used to configure a network.

3.3.1 The One Touch Management System

In this section we will outline our design of a PBNM system. Our design is based on the approach taken by the IETF Policy Group.

Figure 3.1: System Architecture

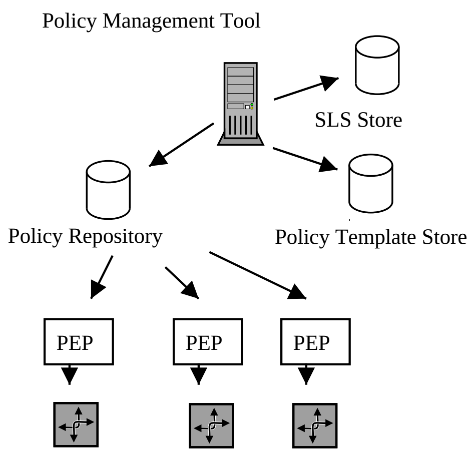
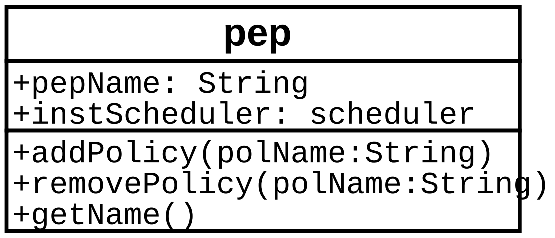


Figure 3.2: UML Diagram of PEP



```
pep
+pepName: String
+instScheduler: scheduler
+addPolicy(polName:String)
+removePolicy(polName:String)
+getName()
```

Policy Enforcement Point

In the design of the policy deployment system each packet-forwarding device that the system configures will be represented by a PEP. When the system user deploys a new policy on to the network it is done through the management tool. The management tool contacts each PEP that the policy is applicable to. The list of PEPs that a policy is to be deployed on to is contained in the policy itself. When the PEP is notified of the existence of a new policy it downloads the policy from the Policy Repository. Then the PEP starts a condition variable, which continuously monitors the current time until the install time is reached (the install time as well as the removal time are part of the policy). The PEP then deploys the policy on to the router. Once this has been done a new condition variable is started which monitors the current time until the time is reached for the policy to be removed. At this time the routers configuration is changed to reflect the removal of the policy. It is important to note that it is not obligatory nor may it be optimal for the PEP to reside on the packet-forwarding device itself. Under our design all that is required is that there would be a one to one relationship between each PEP and each packet-forwarding device.

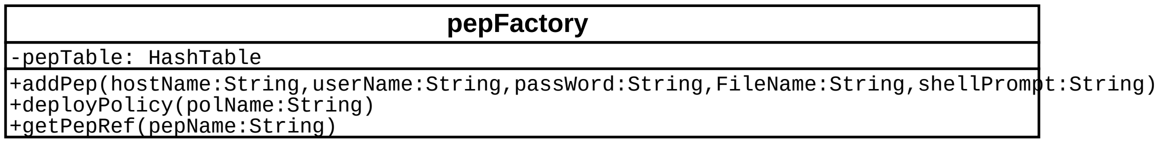
As can be seen by the UML diagram in Fig 3.2 the PEP has two parameters

- *name* the name of the PEP
- *instScheduler* a scheduler object. The scheduler starts and stores all the condition variables.

The PEP has the following methods

- *addPolicy(String polName)* this method is used by the management tool to notify the PEP of a new policy

Figure 3.3: UML Diagram of pepFactory


pepFactory
-pepTable: HashTable
+addPep(hostName:String,userName:String,passWord:String,FileName:String,shellPrompt:String)
+deployPolicy(polName:String)
+getPepRef(pepName:String)

- *removePolicy(String polName)* this method is used by the management tool to notify the `PEP` that a policy should be removed
- *getName()* this method is used to return the name of a `PEP`

pepFactory

Each `PEP` is a logical entity in itself, so as to enforce this concept we have separated the `PEP` points from the management tool. The `pepFactory` object is used to store the `PEP` objects. As can be seen by Fig 3.3 the `pepFactory` has two methods

- *addPep(hostName,userName,passWord,FileName,ShellPrompt)* this is used to add a `PEP` to the `pepTable`
- *deployPolicy(polName)* this method is part of the functionality of the management tool, its purpose is to notify the relevant `PEP` of new policies
- *getPepRef* the third method is used to return a `PEP` object given the name of the `PEP`

The `pepFactory` has a single variable that been the `pepTable`, the `pepTable` stores all the instances of `PEPs`

The `pepFactory` when instantiated is registered with an RMI registry, this enables us to run the policy management tool and `pepFactory` on different hosts.

Policy Management Tool

The main purpose of our policy management tool is to provide a means where the user can create and deploy a policy. In the system a policy can be deployed by the user entering the name of the policy that they wish to deploy, the policy management tool will also provide the functionality where the user can remove a policy that been deployed on to the network also by means of entering the name of the policy they wish to remove. Due to the fact the system is being designed as a matter of research we have not implemented a naming schema for our policies.

Policy Repository

The purpose of the Policy Repository is to provide a storage mechanism for policies. For the purpose of this dissertation we used a single table to store our policies. In this table we stored the following information

- *policy Name* the name of the policy
- *Install Date* the date that the policy is to be installed
- *Install Time* the time of day that the policy is to be installed
- *Remove Date* the date that the policy is to be removed
- *Remove Time* the time of day that the policy is to be removed
- *Router Name* the Name of the router the policy is to be installed on
- *config* the actual policy to be installed on the router

3.3.2 Communications Architecture

In this section we discuss how each component of our system communicates with the other components.

Policy Management Tool and Policy Enforcement Point

The policy management tool needs to contact the `PEP` for the purpose of notifying it of policies that are to be deployed. For this we use Java RMI to communicate the name of the policy that is to be deployed. The `pepFactory` is implemented with the Factory Pattern, this enables the policy management tool to specify the name of the `PEP` it wants to communicate and the `pepFactory` will return it a reference.

Policy Deployment

Due to the fact that the `PEP` does not reside on the router we need to have a method of communicating configuration changes from the PEP to the router itself. There is two parts to this process firstly we must transfer a configuration file from the `PEP` to the router for this we have decided to use the FTP protocol. Once we have the configuration file transferred to the router we need to stop the ALTQ daemon and restart it with the new configuration file, at first glance it would seem a distributed systems technology such as RMI would be ideal for controlling the router, however to install a JVM on a production router would not make sense due to Javas lack of speed. Instead of using Java RMI we returned to a more orthodox communication tool in the form of Telnet. Telnet is used to stop the ALTQ daemon on the router, an FTP client is used to change the configuration file and the ALTQ daemon is then restarted using Telnet.

3.4 Policy Creation

This section will cover how we have defined a policy in our system including what information is part of a policy, from there we go on to discuss different strategies of creating policies and finally we cover in-depth the strategy that has been used in this dissertation for designing policies.

3.4.1 Defining the Concept of a Policy

How to define a policy is a key issue in a PBNM system. Should a policy be restricted to a set of commands which will enforce QoS on routers defined by the Network Administrator, which are not part of the policy or should a policy specify a source and destination points including intermediate routers and optionally different actions for each router. The last option is more in keeping with the concept of configuring a virtual circuit yet this is the type of policy which we come out in favour of. The first option is a simple one but it has the following flaws, firstly if only conditions and actions are defined in a policy it would be unreasonable to attach an SLS with this policy because it can be deployed to many different combinations of source and destinations. The second flaw with this approach is that if a policy is applied widely throughout the network the SLS of that policy would be confined to the worst SLS possible for traffic through the network. In the main it is a strategic decision to limit a policy to a specific set of routers. As mentioned above if we did not limit a policy to a specific set of routers we would be confined to offering the user the guarantee of the worst SLS through the network, This has two important implications

- It is quite possible that the traffic will always be taking the best route through the network but will be paying for the worst route

- The number of policies that we will be able to offer from our network will be reduced, reducing the maximum revenue we can gain from the network.

This is because in this situation where there are two possible ways of traversing the network, if the route the traffic is to take through the network is not specified there would have to be resources reserved on both routes through the network. This has the following implication, instead of being able to offer two SLSs, one for each possible route through the network, there can only be one SLS offer because the traffic could take either of the two routes through the network.

The idea of selecting which routers that a traffic flows is to take through a network is the same idea as is used with MPLS. The difference is that MPLS is used so that traffic spends less time at the router.

For example in Fig 3.4 there are two paths in which the traffic can traverse the network: Path A and Path B. If we had a client who required an SLS between the source and destination we would recognise that without specifying the routers that their traffic could take it is to take we would have to offer him the lowest SLS of either Path A or Path B.

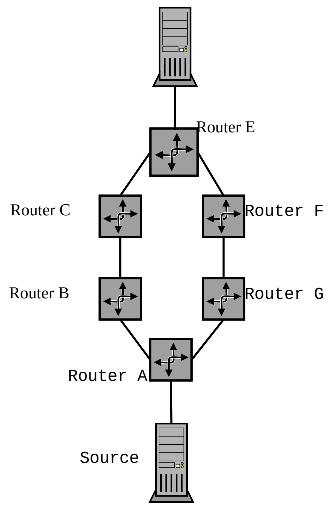
Path A: Router A - Router B - Router C - Router D
Path B: Router A - Router G - Router F - Router E

The policy is the actual commands that are implemented on the routers. The policy will contain four parts.

1. A part which filters the packet, this will take the form of an access control list or a set of QoS CLI command.‘

2. The actual QoS commands which deliver the QoS.

Figure 3.4: Network Traffic



3. The commands which will dictate the next router the traffic is to traverse to.

4. Timing of the installation and removal of the policy.

3.4.2 Evolution of Policy a Creation System

In our examination of the problems encountered in developing an authoring system for policies we have identified five strategies for policy creation. Each strategy has its own usefulness and effectiveness, but also flaws that would prevent it being a realistic method of policy suggestion.

Strategy One

This is the simplest form of policy creation; this involves choosing a system predefined policy and suggesting the time that the policy should be installed. This type of policy creation is a very simple strategy and can only be complicated by considering that other policies may be installed at the same time that the user requires this policy to be installed. This problem could be overcome by creating a resource database, which would keep a log of policies that are confirmed to be deployed at a certain time, in this way the user would know if other policies are already scheduled to be deployed on to the network at the time that the user requires to deploy these policies. Complications arise where we are unsure of the consequences of having two different policies installed on the same router. To avoid the last problem occurring we could make a sacrifice and say that only one policy may be installed on a single router. This strategy is far from optimal but it prevents conflicts between policies occurring.

Strategy Two

The second strategy that we will look at can be also regarded as a simple policy suggestion strategy. This strategy involves authoring a set of policies of a variety of template and using a variety of parameters and then measuring their SLS and storing the result in a database. The user would then enter their SLS and the system would then find the SLS closest to the users requirement. This strategy at a first glance would appear to be a good one yet it is quite limited in the fact that if a perfect match is not found and a policy is suggested with an SLS close to the users requirements, the user will not know how to change the policy so as to suit their SLS and at worst it may be impossible for the user to change the policy to meet their requirements.

Strategy Three

This is third type of policy creation strategy that we have identified, it involves the creation of a set of policy classes with parameters, which can be changed by the user. This type of policy has a major advantage, the user does not need to understand the complexities of policy creation plus the laborious task of creating a policy for each router. However when using this form of policy creation the user does not know the consequences of the policy they created will have in terms of the SLS it will meet. This type of policy creation is of limited use in that the user has created a policy but its effectiveness is a not known. This type of policy creation also has the problem of the user being uncertain as to the type of policy should be used to gain their objectives.

Strategy Four

As we have mentioned above giving a user the opportunity to create policies using their own parameters gives the users a policy creation environment but lacks credibility in that the user has only limited knowledge of the consequences in terms of the SLS that this policy will meet. To create a better environment for policy creation we allow the user to enter an SLS and let the system decide what type of policy that is required. This system will result in the user getting a suggestion as to what type of policy should be used for the required SLS. This type of policy creation has the benefit in that it guides the user as to what type of policy should be used in trying to meet their SLS but still leaves the user in the dark as to what parameters are to be used so that the policy can meet their required SLS.

Strategy Five

We can enhance the last form of policy suggestion by guiding the user to the parameters that should be chosen so as to improve the policy suggestion process. We can guide the user to the parameters that should be chosen by measuring the SLS for the policy with different sets of parameters, this means that the user now knows the effects of different sets of parameters on the SLS. The effectiveness of this strategy depends on the number of sets of parameters that have been measured, when we increase the number of sets of parameters that have been measured the system becomes more effective likewise with a limited number of sets of parameters measured the system will offer little value to the user.

3.4.3 Policy Creation

As has been mentioned throughout this dissertation the creation of policies is the area under research. In overview we have chosen to create policies on the basis of predefined policy templates. We provide a set of templates of policies which we know the SLS that they will meet and offer advice to the user on how a policy may be changed so as to accommodate the users requirements. The first part in the process of creating policies on the basis of an SLS is to gather the SLS from the user. In our system the SLS is gathered from the user through the use of a GUI. The information that is gathered from the user can be categorised into three categories

- "Target" Category
  - The Source IP Address of the traffic
  - The Source Port of the traffic
  - The Destination IP Address of the traffic
  - The Destination Port of the traffic
- "QoS" Category
  - Bandwidth Required by the traffic

  - Delay Required by the traffic
  - Jitter Required by the traffic
  - Loss Required by the traffic

- "Time Category"
  - Install Date
  - Remove Date

Each category has its own importance.

Target Category

To be able to meet this part of the SLS greatly depends on the resources of the network operator. If the source and destination are within the operator's own domain it is much easier to satisfy this part of the SLS than the situation where traffic must travel through other network operators domains to reach the required destination.

QoS Category

Meeting this category is solely down to the capabilities of your network, if your network has not the capabilities to meet an SLS it is a very costly option to expand your network so that it can meet the SLS.

Time Category

Time Category may be deemed to be the easiest of the above categories to meet in terms of the SLS. In comparison to the other two categories it is easier to satisfy the time category because what hinders the complete satisfaction of the time category is the existence of other policies installed on the network, this obstacle can easily be overcome if there is a clear priority of policies or in a commercial world someone is willing to pay more than someone else so as to have their policy installed at their chosen time. On the other hand what prevents the other two categories being met is a lack of resources normally hardware resources, a resource that cannot be changed at will.

3.4.4 Policy Suggestion

Strategy number five is the policy suggestion strategy we have decided to use for our design. Policy suggestion is based on a number of policy templates, we will have x different policy templates, each template of policy will target a different requirement a user could have in terms of a required SLS. For each policy template we will have six policies measured for their QoS statistics. Each of these policies will have different parameters which are associated to the policy template in question. When a user enters an SLS the Policy Advisor will undertake a matching process where the Policy Advisor will decide what type of policy template is appropriate to the users SLS and what parameters should be used for that template. The Policy Advisor will display what type of policy it has recommended ,its parameters, the SLS it meets and what the configuration file for each router that the policy will be applied to will look like. The policy suggested might not meet the required SLS that the user required but in normal circumstances will be close to that which was required by the user. It is expected that a user would use the policy that the system has suggested rather than changing a policy and not being guaranteed its SLS. Though if the user wishes to change the parameters of a policy the system will notify the user of the effects of increasing or decreasing a policies parameters.

3.4.5 Designing our Policy Templates

In this section we explain how each type of template policy delivers upon its SLS offering. The first set of template policies would be regarded as the “easy” policies due to the fact that their SLS are easier to measure than the more complex policies, and also given the fact that their SLS is simpler to understand.

Policy Template One: Restricting Bandwidth

The purpose of this policy is to block a specific type of traffic based on its source IP address, source port, destination IP address and destination port. The SLS for this policy should indicate zero bandwidth for the traffic defined. This policy is implemented through placing a packet dropper on each ingress router for this type of traffic. This type of policy differs from the other type of policies in that it is possible to generate a policy of this type between any client defined source and destination and still be able to guarantee the SLS that the policy will meet.

Policy Template Two: Confining Bandwidth

This policy template is designed to restrict the bandwidth available to a type of traffic when there are other types of traffic traversing the network. For example this template could be used if you required that MP3 would only be allowed on the network whilst other traffic was not using the network.

This template is implemented using CBQ where a flow of traffic is reserved 0 % of bandwidth but is allow to borrow bandwidth when it is available.

Parameters The user will not be able to change any parameters for this class of policy. This policy has the similar characteristic as the above policy in that this type of policy can be automatically created between source and destination because it offers no guaranteed QoS.

Policy Template Three: Reserving Bandwidth

This policy template is used to reserve a type of traffic an amount of bandwidth x; and allows this traffic to use any extra bandwidth when it is available. This type of policy template will be of use when a user requires x amount of bandwidth, and it is optimal for network usage that whatever extra bandwidth is available is used. Unlike the first two policy templates an SLS for this template is only applicable between the source and destination that it has been measured for.

Parameters This policy template will be implemented using CBQ where the class is reserved x amount of bandwidth, for this policy there will be a single parameter which the user will be able to change this being the amount of bandwidth reserved.

Policy Template Four: Low Delay I

This policy template is the first of the more complex policies; the purpose of this policy template is to offer low delay to the traffic that is specified by the policy.

This policy template is implemented using CBQ in the following way, at the ingress router we allow x amount of bandwidth into the network, this x should reflect the amount of bandwidth that the user requires, in the core routers we reserve x+y amount of bandwidth for the traffic. Traffic using this policy experiences lower delay because there is more bandwidth available for this traffic than is needed, hence queues at the router in the network will be smaller and therefore will be delayed less.

Parameters For this policy template there are two parameters, which the user will be able to change, the first parameter is the amount of bandwidth which is allowed into the network, this should be the users target rate of bandwidth, and the second parameter is the amount of bandwidth which is reserved in the core routers.

Policy Template Five: Low Delay II

The purpose of this policy template is also to reduce the delay, which is affecting the traffic specified in this class. This policy works by reserving x amount of bandwidth for this traffic and placing a RED dropper on each queue for this traffic at each router.

Parameters For this policy template there is only one parameter, which may be changed, that being the amount of bandwidth reserved for this type of traffic.

Policy Template Six: Low Loss

The purpose of this policy template is to offer low loss to the traffic that is using it. We implement this policy by placing a token bucket on each applicable router. This algorithm offers low delay because we can store more packets at the router which means in time of congestion that packets are not dropped as fast as they would be without the RED dropper on the router. This policy has the side-effect of increasing the delay experienced by traffic because the packets can now spend longer at each router.

Parameters This policy has only one parameter that being that amount of traffic this policy is to be offered to.

3.4.6 Policy Recommendation

Part of the novelty of this strategy is that the users SLS requirements are not just simply matched to the policy templates, but the users SLS is matched to the closest policy template that matches the users SLS and the system recommends how a policy may be changed so as to reflect how the policy instance may be changed. Given the fact that we have different types of policy templates we had to incorporate a recommendation process into each policy template. Each policy template has a method which accepts two SLSs, one SLS being the users required SLS and the other SLS being the SLS of the recommended policy. For each of the following parameters: bandwidth, delay, jitter, loss the system recommends how the parameters of the policy may be changed so as to meet the required SLS. This recommendation will take the form of a suggestion to increase or decrease a variable. Part of the recommendation will also advise the user on how changing one parameter may effect another QoS parameter apart from the one the user is altering.

3.4.7 Policy Creation

The creation of a policy which can be directly deployed on to a router is the same process for each policy template. The process of creating a policy is undertaken with the following information.

- Router Information
    - IP Address
    - Interface Card Name
    - The link Bandwidth
    - Type of Router (Ingress, Core or Egress)

- Policy Information
  - Policy Type
  - Policy Parameters

Each policy has a method defined to create an ingress policy, core policy and an egress policy. When we ask for a policy to be created the policy accepts the information pertaining to the router that the policy is to be deployed on to. This method has a policy template for the type of policy that needs to be created and matches the information on the router on to the template and returns the policy to the user.

3.5 Architecture in Operation

The operational description of the Architecture is as follows:

1. The Customer approaches the network operator with an SLS that it wants enforced between a source and destination.
2. The network operator enters the customers SLS into the Policy Advisor.
3. The Policy Advisor suggests a predefined policy which matches the customers SLS or one which is close to it.
4. The network operator then offers to enforce the SLS that the Policy Advisor has suggested.
5. The network operator forms a SLA between itself and the customer on the basis of the SLS specification suggested by the Policy Advisor.
6. The network operator deploys the policy on to the network. The management tool is informed of the policy to be deployed. The management

tool downloads the policy from the Policy Repository and contacts each of the relevant PEPs.

7. When the PEP is made aware of a new policy to be installed, it downloads the policy, which is relevant to the specific router that is important to this PEP.

8. When the policy has been downloaded to the PEP it starts a scheduler thread, which continually checks the current time until the time occurs when the policy should be installed.

9. The PEP will configure a configuration file with the policy that was downloaded, if the PEP does not reside on the same host as the router, the configuration file is FTP 'ed on to the router and we use Telnet to restart the daemon with the new configuration file.

10. When the PEP has installed the policy it starts another scheduler, this scheduler is used to remove the policy at the specific time that is specified in the policy.

3.5.1 Information Architecture

In this section we describe the information that is stored to help in the creation of the Policies.

Service Level Specification

As was mentioned is mention in Chapter 2 the SLS is the term used to describe the collection of parameters which define a level of QoS. Our representation of an SLS takes the form of an object with the following fields.

- *Bandwidth* the bandwidth figure

- *Treatment* Defines what level of bandwidth the traffic is given. For example the Treatment could define that the the policy cannot use more than x amount of bandwidth even though it might be available
- *Delay* the delay figure
- *Jitter* the jitter figure
- *Loss* the loss figure
- *Traffic* This variable defines the Transport protocol of the traffic
- *Filter* The filter object is an object which describes the source and destination that this SLS is applicable to.

Policy Store

The Policy Store is used to store our policy templates, in our design we have assigned a vector to store each set of policies for each type of template. If we were to further develop this system we would use a database to store the policy template with a table for each policy template. For example the Table 3.1 shows the policy template four table.

Bandwidth Required	Bandwidth Reserved	Policy code	Source IP address	Source port	Destination IP Address	Destination port	Traffic protocol

Table 3.1: Policy Four Template

SLS Store

The purpose of the SLS Store is to provide a means of storing the SLS objects.

3.6 Policy Language

The policy language that was used in this system was the policy language which is supported by ALTQ [Cho] . ALTQ is a package that provides Alternative Queuing for BSD UNIX developed at the Sony computer laboratories in Japan.

ALTQ is designed as a research tool for QoS in IP networks, and not as a tool that would be consider for deployment in a production network, partly due to the fact that to change the configuration of the router the daemon must be killed and any data in the buffers would be lost. ALTQ supports the following queuing schemes [Cho]

- First in First Out (FIFO)
- Weighted Fair Queuing (WFQ)
- Random Early Detection (RED)
- RED with In and Out bit (RIO)
- Class Based Queuing (CBQ)

ALTQ operates through the use of six daemon programs: fifod, wfqd, redd, riod, cbqd and rsvp, these daemons provide the required queuing behaviour. A limitation of the package is that only one daemon may be in operation on a network interface at any one time.

To change the configuration of an ALTQ router you must change the configuration file belonging to the router. As is mentioned above ALTQ prevents multiple queuing disciplines being in operation at the same time. This means that the policies we created could not be as complex as they would be if we could have used multiple queuing algorithms installed on the same router.

In appendix A we show two ALTQ configuration files.

3.7 Summary

In this chapter we have examined a design for a policy deployment system. Each component of the policy deployment system was examined from a design point. The next part of the chapter was concerned with the design of a policy creation system, different strategies for composing policies were examined. The policy creation system that was chosen is based on a matching and recommendation process. Finally we examine the architecture of the policy creation system.

Chapter 4

Implementation

4.1 Introduction

This chapter discuses the implementation details of the framework that was developed for this dissertation.

4.2 Policy Deployment

The issue of how we would actually deploy a policy on to the routers was a concern of ours from the start of the dissertation. Normally network administrators would Telnet a router to change the configuration of the router. We solved the problem of policy deployment using two forms of communication. To transfer the policy to the router we used FTP, instead of having to develop an FTP client an open source Java FTP client [Ltd] was used. Once the configuration file was placed on the router an open source Telnet client was then used to change the configuration file [JM]. First the process of transferring a file to the host is outlined.

```
FTPClient ftp = new FTPClient(hostName,21);
```

```
ftp.login(userName,passWord);
ftp.chdir(directory);
ftp.put(localPath,remoteFile);
ftp.quit();
```

The first line of code creates a new `FTPClient` object which will connect to the host as using portnumber 21, the second line logs on to the host with the userName and hostName as specified. The next line of code changes the client to the directory that the file is to be place, the file is place on the host, localPath denotes the name of the file that should be transferred, whilst the remoteFile is the file which is to be replaced on the host. After the file has been transffered then close the FTP connection

The second process is involved in changing the configuration of a router is to stop the ALTQ daomon and restart it with the new configuration file. Here is the process by which restarted the ALTQ daomon is restarted.

```
TelnetIO tio = new TelnetIO();
tio.connect(hostName)
wait("login:");
send(userName + "\r");
wait("Password:");
send(passWord + "\r");
wait(shellPrompt);
send();
```

In the above code we have a `TelnetIO` object which is our Telnet client. The first two lines of code is to specify the host to connect to. Then we instruct the telnet client to wait until the login command has been transmitted, at that stage we enter the user Name, we then do a similar process for the password. Once we have entered the password we wait for the shell prompt, the shell prompt has been specified by the user. We then transmit the commands to change the configuration file.

4.2.1 Policy Enforcement Point

As is mentioned in our design chapter each packet forwarding device is represented by a `PEP`. For a packet forwarding device to be represented by a `PEP` we must have a method for associating a `PEP` with a packet forwarding device we do this by creating a GUI which will enable the user to enter the information which is necessary to create the `PEP`. In our architecture it must be possible for the Policy Management Tool to contact each `PEP`. The `PEP`s are stored in a `pepFactory` object. This `pepFactory` provides two functions the first function is the ability to a add `PEP` to the `pepFactory`, the second function provided by the `pepFactory` is the ability to return a `PEP` object based on its name.

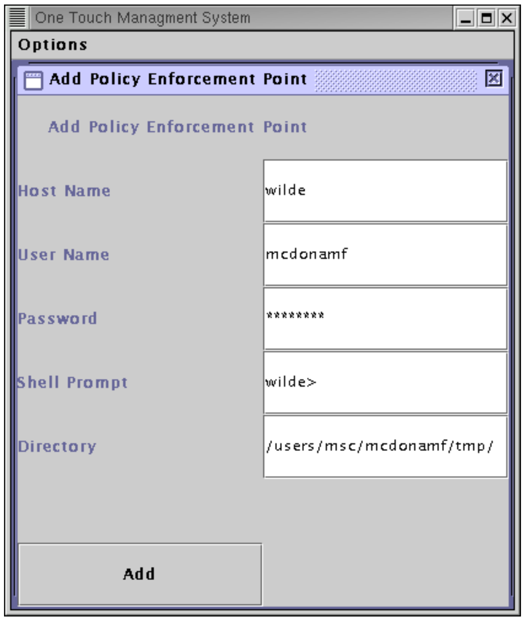
4.3 Matching Algorithm

In this section we will outline how we match an SLS to a policy. This process has two parts to it, the first part is to match an SLS to the correct policy template and the second part is to find the correct policy from the template.

4.3.1 Finding the Correct Template

Finding the correct template is the first part of the matching process. Then we match an SLS to a policy template by examining the contents of the SLS. For an SLS to be matched to a policy it must contain a `filter` object plus

Figure 4.1: The Policy Enforcement GUI


One Touch Managment System
Options
Add Policy Enforcement Point
Add Policy Enforcement Point
Add

Host Name	wilde
User Name	mcdonamf
Password	********
Shell Prompt	wilde>
Directory	/users/msc/mcdonamf/tmp/

one or more of the QoS fields should be full. The following line of code shows how a policy template is chosen

```
if(( slsMatch.getBandwidth() !=0) && (slsMatch.getDelay() !=0)){
```

Once we have found the policy templates which match the type of policy required by the user we need to find the correct policy instances of that template that match the users SLS.

4.3.2 Finding the Correct Policy Instance

Now that we know which template matches the users requirements we have to find the policy instances. If we return to the example of the policy template which offers low delay and we need to match the SLS to the policy. The first part of the matching process is to find all the SLSs which have the same source and destination as the SLS required by the user. Then we find the SLS instances with a bandwidth equal to the bandwidth required, greater than the bandwidth required and the SLS with bandwidth less than what is required. It may seem strange that we pick our SLS on the basis of bandwidth variable instead of the delay variable, in our opinion a users primary requirement will always be bandwidth. For example if a user requires an SLS for an IP video conference and had a requirement that includes a bandwidth and a delay parameter meeting the bandwidth parameter would be of higher priority than meeting the delay requirement.

4.4 Creating a Policy

In our PBNM system we can deploy a single policy on to the network, and it will configure multiple routers, in this section we will outline how we create a policy which can be deployed on to multiple routers. Here we will outline the different parts to the `lowDelay` policy. The `lowDelay` policy is an implementation of the policy `abstractPolicy`. `abstractPolicy` has a three variables.

```
protected routerVector nodes;
```

This variable holds a `router` entry for each router that the policy is to be deployed on. The second variable that `abstractPolicy` has is a `filter` object,

```
protected filter polFilter;
```

The `polFilter` object is used to to define what traffic the policy is to be targeted at. The `polFilter` object will define the source IP address, the source port address, the destination IP address, the destination port address and the protocol of the traffic.

The third part of the abstractPolicy is the code of the policy, this is simply a unique number which is assigned to a policy to identify it. This is the part of the policy which is specific to the lowDelay policy. The following are the variables belonging to the lowDelay policy which are unique to the lowDelay policy.

```
protected double bandReq;
protected double bandRes;
```

This bandReq variable represents the bandwidth that is allowed into the network at the ingress router whilst the bandRes variable is the variable which represents the amount of traffic that is to be assigned to this class at the core routers. We use the following method to create a policy

```
public String createPol(String routerName);
```

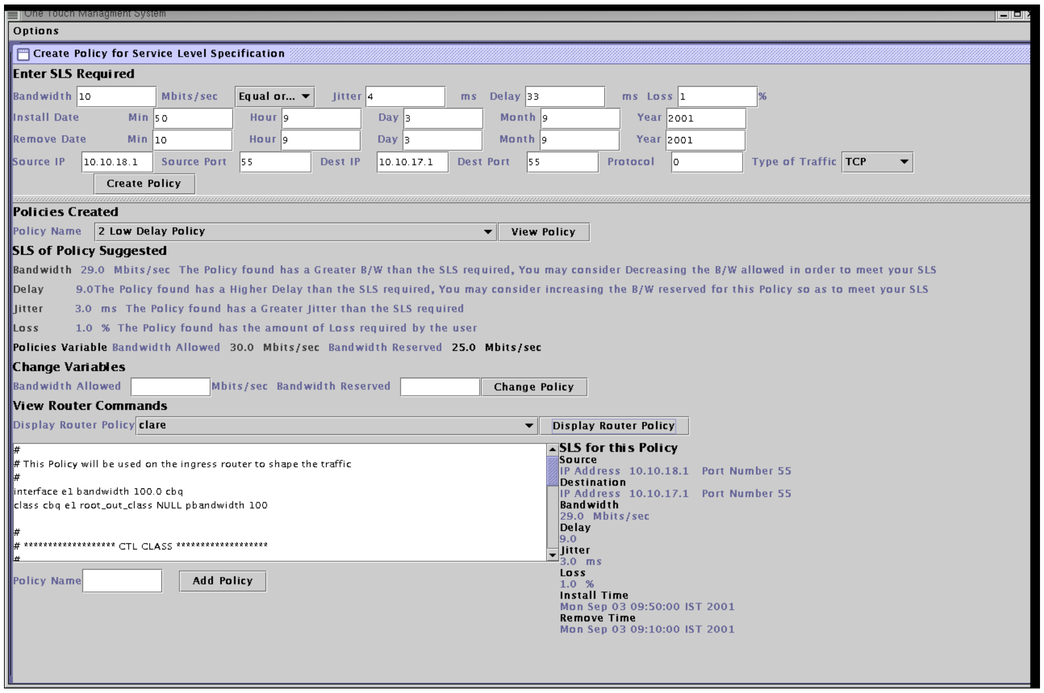
This method then gets the router object from the routerVector, it then decides what type of policy is to be created and calls one of the following methods

```
public String createIngressPol(String interfaceName, double linkBand)
public String createCorePol(String interfaceName, double linkBand)
public String createEgress(String interfaceName, double linkBand)
```

Depending on the type of router one of the above methods will be used to create a policy that can be placed on a router. In each of the methods the interfaceName variable represents the name given to the interface card on the router. The linkBand variable represents the bandwidth available on the connection into the network interface card.

In the creation of the policy we have a predefined set of commands which we insert the name of the network card, the bandwidth of the link, the traffic

Figure 4.2: The Policy Advisor GUI



Create Policy for Service Level Specification

Enter SLS Required

Bandwidth 10 Mbits/sec Equal or... Jitter 4 ms Delay 33 ms Loss 1 %

Install Date Min 50 Hour 9 Day 3 Month 9 Year 2001

Remove Date Min 10 Hour 9 Day 3 Month 9 Year 2001

Source IP 10.10.18.1 Source Port 55 Dest IP 10.10.17.1 Dest Port 55 Protocol 0 Type of Traffic TCP

Policies Created

Policy Name 2 Low Delay Policy

SLS of Policy Suggested

Bandwidth 29.0 Mbits/sec The Policy found has a Greater B/W than the SLS required, You may consider Decreasing the B/W allowed in order to meet your SLS

Delay 9.0The Policy found has a Higher Delay than the SLS required, You may consider increasing the B/W reserved for this Policy so as to meet your SLS

Jitter 3.0 ms The Policy found has a Greater Jitter than the SLS required

Loss 1.0 % The Policy found has the amount of Loss required by the user

Policies Variable Bandwidth Allowed 30.0 Mbits/sec Bandwidth Reserved 25.0 Mbits/sec

Change Variables

Bandwidth Allowed Mbits/sec Bandwidth Reserved

View Router Commands

Display Router Policy clare

```
#
# This Policy will be used on the ingress router to shape the traffic
#
interface e1 bandwidth 100.0 cbq
class cbq e1 root_out_class NULL pbandwidth 100

#
# ****************** CTL CLASS ******************
#
```

Policy Name

SLS for this Policy
Source
IP Address 10.10.18.1 Port Number 55
Destination
IP Address 10.10.17.1 Port Number 55
Bandwidth
29.0 Mbits/sec
Delay
9.0
Jitter
3.0 ms
Loss
1.0 %
Install Time
Mon Sep 03 09:50:00 IST 2001
Remove Time
Mon Sep 03 09:10:00 IST 2001

profile defined by the `policyFilter` and finally the `bandReq` or the `bandRes`. The `bandRes` will be expressed as a percentage of the bandwidth of the link. Hence the same for two core routers will differ in their interface card name and also the percentage of traffic `bandRes`.

4.5 The Policy Advisor GUI

Fig 4.2 shows the GUI which we have implemented for our policy creation environment. The GUI can be divided into three parts. The first part of the GUI is the *Enter SLS Required* section. It is provided for the user to enter the SLS that they require. There is a section for each variable to be entered.

The second part of the GUI is the *Policies Created* Part. This part of the GUI has a dropped down list, the items in the list are the policies which have been suggested to the user. If a user wants to see a suggested policy they must select the policy they wish to see and click the View Policy Button.

The third part of the GUI is the *SLS of Policy Suggested* part. The purpose of this part of the GUI is to show the user the SLS of the Policy that has been suggested and the parameters of this policy. Not only is the SLS of the policy shown in this section but we also show the recommendations of the Policy Advisor for how to change the policy so as to meet the required SLS.

The fourth part of the Policy Advisor GUI enables the user to change the parameters of the policy that has been recommended. For each type of policy that can be recommended there are different fields which can accept parameters appropriate to the policy.

The final part of the Policy Advisor enables the user to see more of the details of a policy and its accompanying SLS. Though the parameters of a policy maybe be enough information for some network operators others will wish to see the actual configuration file which will be placed on the router. To allow for this we have a drop down list of all the routers a policy will be deployed to, and the user can see the configuration file of each router. There item is a button which enables you to add your policy to the Policy Repository.

4.6 Summary

This chapter has discussed the implementation of the policy deployment and creation environment that was created for the dissertation. The approach taken to matching SLS to policies was examined in detail as was the systems GUIs.

Chapter 5

Evaluation and Conclusions

5.1 Introduction

The objective of this chapter is to evaluate the design of our policy deployment system and also our policy creation environment. The next part of the chapter will concern itself with the conclusions of the dissertation and finally we outline the scope for further research in the area.

5.2 Objectives Revisited

Before we start to evaluate the work of this dissertation it is important to remember to objectives of the thesis.

- Research the different approaches to QoS, inparticular the PBNM approach to QoS.

- Investigate the different issues and challenges involved in the semi automatic authoring of policies for different SLAs.

- The final object in our research was to develop a prototype system for policy deployment and policy creation.

5.2.1 Approaches to QoS

Through our research we have come to the conclusion that DiffServ will become the QoS protocol call of choice for backbone networks. DiffServ is appropriate for routers on backbone router because it requires little configuration and also once DiffServ has been enabled on a router it does not require further configuration. For speed requirements it would be possible to implement DiffServ in the hardware of a router. PBNM will be the appropriate protocol for corporate networks in that it can offer a range of management options aside from bandwidth reservation which is the only service IntServ offers. DiffServ would be suitable for a corporate network but it does not offer the ability to create the wide range of policies that PBNM allows. MPLS is more of protocol optimising an IP network, though we do incorporate some of the ideas of MPLS in our policy creation environment.

5.3 Evaluation of Policy Deployment System

The policy deployment system which has been developed for this dissertation will be in evaluated by comparing it to the Policy Schema as defined by the IETF. This is an appropriate method of evaluation because it enables us to evaluate our policy deployment system whilst at the same time evaluating the IETF standard as well. In evaluating our system we will examine each component separately.

5.3.1 Policy Management Tool

The Policy Management Tool we have implemented is a very much a scaled down version of Policy Management Tool as specified by the IETF. Our Policy Management Tool is provided to enable a user to deploy a policy by entering the name of the policy. We have omitted the following functions from the Policy Management Tool as specified by the IETF

- A Policy Translator
- A Policy Validation Tool
- A Global Detection Function

The purpose is to of the Policy Translator is to translate general policy specifications into policies, the system developed has this facility in that a user can specify policy parameters and they can be translated down into policies. A Policy Validation tool would be a useful addition to the system but was not implemented due to time constraints. A Global Detection function was not needed as only one policy can be installed on a router at any one time, what may have being useful is a tool that would detect whether another policy is to be installed at the same time the user wishes to deploy their policy.

5.3.2 Policy Repository Tool

The IETF Policy Repository has the purpose of storing policies, it is implemented as an LDAP directory, our Policy Repository is represented as a database. A database lacks the fast look up speeds that a directory offers. Another point of note is that in the IETF policy model PEPs poll the Policy repository for the existence of new policies. In our system we take a different strategy the management tool notifies the PEPs of the existence of new policies, this avoids the unnecessary polling and also only the PEPs that a new policy is to be deployed on are notified of the new policies.

5.3.3 Policy Enforcement Point

In our system the role of the Policy Consumer is undertaken by the Policy Enforcement Point, as you can recall the Policy Consumer has three goals

- to download policies from the policy repository.
- translate policies into a form which are suitable for a policy target (i.e. a router).
- to deploy the policies on to the policy targets.

Our PEP achieves two out of the three goals of the policy consumer. Our PEP is unable to translate policies akin to what is required by the Policy Consumer, the IETF Policy Consumer needs to be able to translate policies because it is foreseen that there would be a standard policy language for creating policies. When a Policy Consumer receives a policy it would have to translate the policy into vendor specific commands. Currently there is no standard policy language so we did not forsee the need of incorporating a policy translation function into our system, a disadvantage of this is that our framework cannot support heterogeneous devices.

5.3.4 Strengths and Weakness of Framework

A disadvantage of our architecture is the fact that only one policy may be deployed on to the router at any one time. An improvement of the system would be where we could deploy numerous policies on to one router. The ideal situation would be if we could deploy a policy on to a router at time x and deploy a different policy on to the same router at time x + y, whilst being able to remove a policy from a router without removing the other policies on the router. To do so two issues of concern must be overcome one how to deploy different policies on to a router without them conflict in terms of their aims, and two how to remove a policy from a router without damaging the other policies which are installed.

An important part of evaluating the framework is to highlight the weakness in the framework. Highlighting the weaknesses in the framework is important because it place the advantages of the framework into perspective and secondly it highlights avenues where the framework could be improved.

The second disadvantage of the architecture is the scenario where we have a network which is being expanded, if the resources in the network are changed our SLSs are no longer valid. This would mean each policy instance in the system would have to be re-measured. This is a disadvantage but in any type of management system which provides SLSs a change in network configuration will invalidate the SLS.

5.3.5 Policy Authoring

In this section we will evaluate the policy creation environment that we have created. Before we evaluate our Policy Authoring it is is important to recap the design approach we took to our policy authoring system. The design approach taken was a template approach, the template approach separates the policy into parts, the first part of template is the policy specific parts and the second part of a template is the parameters a user may alter.

The main advantage of our architecture is that it enables network operators to focus on the parameters of a policy instead of the different configurations of a policy. Without a system such as the one created for this project a network operator would have to spend a greater amount of time authoring policy syntax than changing the parameters of a policy. This means that a network operator will be able to concentrate more on the parameters of a policy, this also has the side effect that the network operator will develop a better understanding as to how the parameters of a policy effect the level of QoS deliver.

The second advantage of our system is that the system can advise a user as to the type of policy and parameters of that policy that can meet an SLS, without such a system a network operator would use a trial and error method of finding a policy to meet an SLS or use a policy which has an SLS higher than the one required. Another side effect of the recommendation of the system is that the network operator learns which type of queuing algorithms will deliver different types of QoS

The third advantage of our system is that if the system cannot suggest a policy with the exact SLS required it will suggest a policy that has an SLS close to the required SLS. The system also advices the user on how a policy may be changed so as to meet an SLS, this means that if a policy does not meet the required SLS the user has the option of changing the policy.

The fourth advantage of our system is that a policy is deployed on to a specific set of routers, the importance of this can be seen by the Fig 3.4. In a scenario where we do not specify the routers the traffic is to flow through. The traffic can take either Path A or Path B, this means that the SLS for this policy will be worst of the SLSs of the two paths another disadvantage of this approach is that we have more policies than is possibly needed. If we define the route we can now offer two SLSs through the network to two different users with the effect of deriving more value from the network.

The last advantage stresses the relationship between a policy and its SLS.

The problem of authoring policies is to have them reflect SLSs. It is not as difficult a task to create an automated system that can create policies as to create an automates system which may create policies on the basis of an SLS.

One of the main perceived disadvantages of the policy creation environment is the fact that a policy can only be suggested for a predefined source and destination addresses. We acknowledge that this is a constraint on the system but to offer an SLS with an assurance, the SLS between a source and destination must be measured. If an SLS is going to be part of an SLA which specifies high monetary penalties if the SLA is broken a user must be sure that the policy that has been recommended will actually meet the SLA required

Other Policy Creation Issues

An other point of note in relation to our policy creation is the fact that the install time and remove time of a policy can be dictated by the user, this is unlike the other QoS variables which can only be decide by the system.

5.4 Conclusion

5.4.1 Policy Authoring

An improvement to the architecture would be the addition of a resource database to the policy suggestion system. The purpose of the resource database would be to record the times different policies are to be installed. With this resource database we could prevent the install times of two policies overlapping.

An area of further research for this dissertation would be the use of traffic engineering so as to ensure the guarantee of an SLS.

A second area of further research is the creation of a standard Policy Language, this would enable the development of PBNM considerably yet the development of a standard policy language would need the backing of the major networking vendors who may see it more profitable for them to only support their own propriety policy language.

A third area of research is the guarantee of an SLS, how do we guarantee to a user that a policy will meet the required SLS.

5.4.2 Final Conclusions

The conclusion which can be drawn from this dissertation are as follows

- DiffServ will be used on backbone networks, PBNM will be used to manage small corporate networks.
- Role Based Management could reduce the management needed to manage small corporate networks.
- The system designed eases the task of policy creation by
  - providing a GUI which abstracts the policy syntax.
  - allow the user to focus on policy parameters.
  - enable the user to select the time for a policy to be installed and removed.
- if an SLS is to be guaranteed, the route the traffic takes should be selected, this means we can deploy more policies on to the network

Bibliography

[BBC$^+$98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. RFC 2475:an Architecture for Differentiated Services, December 1998. Status: PROPOSED STANDARD.

[BZB$^+$97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. RFC2205: Resource Reservation Protocol (RSVP), September 1997. Status: PROPOSED STANDARD.

[Car99] Jeff Caruso. Policy Madness. `http://www.nwfusion.com/newsletters/lans/0426lan1.html`, April 1999.

[Cho] Kenjiro Cho. Managing Traffic with ALTQ. `http://citeseer.nj.nec.com/cho99managing.html`.

[Com01] Hewlett-Packard Company. *HP OpenView PolicyXpert, User's Guide Edition 1*, second edition edition, March 2001.

[DDCH$^+$00] Ed.and J. Boyle D. Durham, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. RFC 2478:the COPS (Common Open Policy Service) Protocol, January 2000. Status: PROPOSED STANDARD.

[Eur00] Eurescom. State of the Art of IP Inter-domain Management and Supporting Measurements. `http://www.eurescom.de/`

public/projectresults/P1000-series/1008d1.asp, July 2000.

[FJ93] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE ACM Transactions on Networking*, August 1993.

[Inca] Cisco Systems Inc. Cisco QoS Policy Manager. http://www.cisco.com/warp/public/cc/pd/wr2k/qoppmn/index.shtml.

[Incb] Cisco Systems Inc. Modular Quality of Service Command-Line Interface. http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr%/fqos_c/fqcprt8/index.htm.

[Incc] Orchestream Inc. Orchestream Service Activator. http://213.35.38.196/products/orchestream/default.asp.

[JM] Matthias L. Jugel and Marcus Meibner. The Java Telnet Application. http://www.mud.de/se/jta/.

[Ltd] Enterprise Distributed Technologies Ltd. Java FTP client library. http://www.enterprisedt.com.

[Lup98] Emil Constantin Lupu. *A Role-Based Framework for Distributed Systems Management*. PhD thesis, University of London, 1998. Available from http://www.doc.ic.ac.uk/˜ecl1/.

[MS93] Jonathan D. Moffett and Morris S. Sloman. Policy Conflict Analysis in Distributed System Management. *Journal of Organizational Computing*, 1993.

[Pac] Hewlett Packard. policyxpert. `http://www.managementsoftware.hp.com/products/policyexpert/index.asp`.

[RCD00] R. Rajan, E. Celenti, and S. Dutta. Service Level Specification for Inter-domain QoS Negotiation, November 2000. Status: WORK IN PROGRESS.

[RVC01] E. Rosen, A. Viswanathan, and R. Callon. RFC 3030:Multiprotocol Label Switching Architecture, January 2001. Status: PROPOSED STANDARD.

[Slo] Maurice Sloman. The Policy Framework. `http://www-dse.doc.ic.ac.uk/Research/policies/PolicyFramework.html`.

[ST] John Sikora and Ben Teitelbaum. Differentiated Services for Internet2. `http://www.internet2.edu/qos/may98Workshop/html/diffserv.html`.

[SWM+99] M. Stevens, W. Weiss, H. Mahon, B. Moore, J. Strassner, G. Waters, A. Westerinen, and J. Wheeler. Policy Framework, September 1999. Status: WORK IN PROGRESS.

[Wro97] J. Wroclawski. The use of RSVP with IETF Integrated Services, September 1997. Status: PROPOSED STANDARD.

[WSS+01] A. Westerinen, J. Schnizlein, John Strassner, Mark Scherling, Bob Quinn, Shai Herzog, An-Ni Humynh, Mark Carlson, Jay Perry, and Steve Waldbusser. Terminology for Policy-Based Management, July 2001. draft-ietf-policy-terminology-04.txt.

Appendix A

Appendix

A.1 Router Configurations

```
#
# This Policy will be used on the ingress router to shape the traffic
#
interface e1 bandwidth 100.0 cbq
class cbq e1 root_out_class NULL pbandwidth 100

#
# ****************** CTL CLASS ******************
#
class cbq e1 CTL_class root_out_class pbandwidth 10 control



#
# ****************** DEF CLASS ******************
#
```

```
class cbq e1 DEF_class root_out_class pbandwidth 90 default

#
# ****************** Leaf CLASSES ******************
#

#
# ****************** SHAPER CLASSES ******************
#
class cbq e1 TARGET_class DEF_class pbandwidth 12.0
filter e1 10.10.17.1 55 10.10.18.1 55 0
```

Configuration file for an Ingress Router running ALTQ using a Low Delay I

```
#
# This Policy will be used in the core routers to offer low delay to the
# traffic, it offers low delay be cause it has x bandwidth reseverved for
# it using CBQ, and it only allows x-y into the network
#
# This Policy will be used on the ingress router to shape the traffic
#
interface e2 bandwidth 80.0 cbq
class cbq e2 root_out_class NULL pbandwidth 100

#
# ****************** CTL CLASS ******************
#
class cbq e2 CTL_class root_out_class pbandwidth 10 control
```

```
#
# ****************** DEF CLASS ******************
#
class cbq e2 DEF_class root_out_class pbandwidth 90 default


#
# ****************** Leaf CLASSES ******************
#


#
# ****************** SHAPER CLASSES ******************
#
class cbq e2 TARGET_class DEF_class pbandwidth 14.0
filter e2 10.10.17.1 55 10.10.18.1 55 0
```

Configuration file for a Core Router running ALTQ using a Low Delay I policy