Dual Degree Thesis on

Mapping Application QoS to Network Configurations for DiffServ over MPLS Networks

By

Sudeep Goyal

Roll No. 00D05013

under the guidance of

Prof. Umesh Bellur and Prof G. SivaKumar

Department of Computer Science and Engineering

Indian Institute Of Technology

Bombay

2004-05

Contents

List of Figures

Abstract

The need to provide differentiated levels of service for e-commerce and other applications over the internet has led to the advent of Network technologies such as MPLS and Diffserv which allow us to shape traffic over a given network. However, translating the needs of an application in terms of network resources to specific network configurations still remains difficult and the task is accomplished mostly in a manual way. This is not scalable when there are many applications demanding different QoS levels of the network and these needs are changing dynamically. Our effort is focused on algorithms that can automate the translation of application needs to network configurations in MPLS networks.

Chapter 1

Introduction

1.1 Introduction

The Internet has grown rapidly in last decade. Its growth has both posed opportunities and threats to many business activities. It has become a global channel of communication to include all businesses. The internet is expected to become a medium of communication that would provide a differentiated level of service on the same internet infrastructure. Thus, this differentiated level of service would include data and voice communications that should also be able to provide real and reliable network services as well. In order to provide many services to the same infrastructure network, IETF has proposed many QoS technologies like DiffServ, IntServ and MPLS.

Presently, there exists 2 different levels at which QoS is specified with no mapping between them. Application level QoS which are specificed as SLAs (Service Level Agreements) are specified and usually dealt with only at the application level. Network lower level details which are specified in terms of jitter, delay, throughput are dealt separately with at the network layer and it is usually, difficult for application developers to specify their network level QoS requirements at design or run-time.

In this work, we focus our attention to enterprise networks and enterprise applications. Internet is too big a domain to start tackling, even though the protocols may be the same.

SLA and QoS parameters at the application layer reflects the quality of service to the customer. They can be different according to the types of service. On the other hand, network QoS parameters means the basic metric of performance measurement of network level QoS at the network management layer.

The goal of our effort is to map Network performance metrics (NPM) to QoS parameters (SLAs) and then giving the view of the network in terms of SLAs to applications, pricing, conformance etc and not in terms of NPMs to customer, network provider and application provider. Here, we collect NPMs from the network through SLA agent and then map them to QoS.

Say, some application wants a delay of less than 50 msec. We should be able to collect the Network state (which is nothing, but NPMs which are: traffic congestion, throughput, availability and latency).

As an example for the IP enterprise network, the SLA of some service provider for some application (like video, voice) with the customer can be as follows: The availability over 99.99 % should be assured. The average RTT (round trip time) should be less than 50 msec. The delivery ratio should be more than 98.0%. The QoS network parameters would be in terms of availability, delay, jitterand loss. The network monitoring methods use to measure network performance and QoS parameters can be : active monitoring, passive monitoring and using standard management protocols like SNMP. A service class can be mapped to many NPM. This mapping depends on the type of services and can be very complicated. And the quality information of service should be presented in customer friendly form, not NPMs. Therefore it is necessary to translate the measured NPMs to SLA parameter.

In this report, SLA is used interchangeably with QoS parameters at the application layer. And the QoS parameters at the network level is used interchangeably same as NPMs (network performance metrics).

1.2 Quality of Service (QoS)

As said in [1], QoS is a vaguely defined term. But, here in our context,we are defining QoS as a 'predictable traffic quality' that satisfies following criterion:

1. Delay
2. Jitter
3. Packet loss percentage
4. Throughput

Delay ensures the maximum delay bound that is acceptable to the application. Jitter reflects variation in delay and is important for real time applications. Packet loss probability shows the probability of the packet loss. Throughput is the data rate that the system can transfer. This is an important factor in streaming applications.

1.3 Service Level Agreements (SLAs)

Networks are currently evolving from a best effort only service towards a service that supports different levels of Quality of Service (QoS). The service provider makes a (legally binding) commitment to deliver those specified levels of QoS. The next step is to enable customers to influence the behavior and configuration of their own instance of the service. This is called Customer Service Management. A key concept to enable customer servicemanagement is the concept of a Service Level Agreement.

[2] explains SLAs as: "Bilateral SLAs can also be defined among pairs of organizations that have a symbiotic relationship. In such case each organization has both roles at the same time: it is the provider of its own service and the customer of the service of the other organization. The SLA constitutes the legal foundation for the delivery of the service. SLAs are used by both parties involved; the service provider uses it to have a definite, binding record of what is to be provided. The provider can use this record in case of disputes with the service customer. This also works the other way around: the customer also uses the SLA as a legally binding description of what the provider has to deliver". A SLA typically has the following components [2].

1. A description of the service that is to be provided.
2. The expected performance of the service.
3. A detailed procedure for handling problems with the service.
4. A procedure for monitoring and reporting the service level to the customer.

5. The consequences of the service provider not meeting the agreed service level.

6. A description of under which circumstances the SLA does not apply.

1.4 QoS control Mechanism of Network Elements

This section explains the mechanism to allow QoS control at each network element.

1.4.1 Packet Classifier

This is used to classify packets based on some predefined rules. As determined by the IETF DiffServ Working Group, packet classifier can be of 2 types: Multi-field (MF) classifier and Behavior Aggregate classifier (BA). MF classifier is used at the edges of the network and BA is used at the core network to ensure scalability.

- **Multi-Field Classifier:** This classifies based on the combination of one or more header (like source/destination address, DS field, protocol ID) information and input interface information.

- **Behavior Aggregate (BA) Classifier:** It classifies packet based only on their DS value.

1.4.2 Traffic Conditioner

Traffic Conditioner measures the input traffic and assures that the packet behavior follows the predefined profiles. It performs following functions: metering, marking, shaping the incoming flow and dropping out-profile traffic.

1.4.3 Scheduler

The scheduler controls the packet transmission sequence from queues of individual classes to provide traffic of each class with the QoS level appropriate to PHB of the respective class. The scheduling algorithms are grouped into two major categories: Priority Based Scheduling and Bandwidth-Sharing Scheduling.

1.4.4 Queue Management

Queue control can accurately control packets (including packet discarding) during congestion. As the Queue Control uses a packet priority discarding mechanism, it can differentiate traffic and provide different loss and delay characteristics to the packets belonging to different service classes of the same queue. The common queue control systems include the Random Early Detection (RED).

When the queue length reaches the preset threshold, the RED randomly discards packets and this discard rate is decided by the function of queue length. Packets are discarded in proportion to the flow before the tail packet discarding occurs due to queue overflow.

1.5 Basic Network QoS Technologies

This section explains basic QoS technologies that are available which are IntServ (RSVP), DiffServ, MPLS.

1.5.1 IntServ Networks

IntServ is the architecture that reserves the flow all along the network end-to-end as requested by the application or user and thus, provides guaranteed QoS. RSVP is the signalling protocol that requests bandwidth and other resources for the IntServ architecture. Intserv defines 2 service models: Guaranteed Service and Controlled load. Guaranteed Service is for applications requiring strict QoS and provides the mathematical upper limit on the queuing delay. Control load controls load using multiplex statistics and for applications with higher flexibility that guaranteed service. [11] gives a good overview on Intserv and RSVP. More details on IntServ and RSVP can be found out from [12], [13], [14], [15] and [16].

1.5.2 DiffServ Networks

There is a difficulty in implementing and deploying Integrated Services and RSVP and there are critical issues like scalability in IntServ. And to overcome these issues and to assure simplicity, Differentiated Services (DS) was introduced.

Customers can mark DS fields of individual packets to indicate the desired service or have them marked by the leaf router based on MF classification.

At the ingress of the ISP networks, packets are classified, policed and possibly shaped. The classification, policing and shaping rules used at the ingress routers are derived from the SLAs. The amount of buffering space needed for these operations is also derived from the SLAs. When a packet enters one domain from another domain, its DS field may be re-marked, as determined by the SLA between the two domains.

Thus, the DiffServ network lets an application or flow packets to be marked at the edge router which thus, can be differentiated from other packet flows and thus, differentiated service can be provided. These marked packets are then provided forwarding priority preferences in the core routers depending on the mark put on them by the edge routers.

1.5.3 MPLS (Multi-Protocol Label Switching)

MPLS is a forwarding scheme that has evolved between layer 2 and layer 3 in the OSI seven-layer model. It has evolved from Ciscos Tag Switching.

Each MPLS packet has a header containing a 20-bit label, a 3-bit Class of Service (COS) field, an 1-bit label stack indicator and an 8-bit TTL field. The MPLS header is encapsulated between the link layer header and the network layer header. A MPLS capable router, which is known as Label Switched Router (LSR), examines only the label in forwarding the packet. The network protocol can be IP or anything.

MPLS sets up labels across the LSRs by using some protocol known as Label Distribution Protocol (LDP), which may not be the same in all LSRs. This LDP is used to set up Label Switched Paths (LSPs). A LSP is similar to ATM virtual circuit and is uni-directional from sender to receiver. LSP set-up can be control driven or data driven. In control-driven LSP setup, label distributions are triggered by control-driven traffic like routing updates. In data-driven set up, the label distribution is triggered by request of a flow or an aggregation of flows (known as traffic trunk). LSP set-up between 2 hosts can be done explicitly. Then the path is known as explicit path (EP). This is one of the most useful feature of MPLS and is heavily used in Traffic Engineering (TE). Thus, by using the labels as a result of label distribution during set-up of a LSP, forwarding table is used to forward or process packets based on their MPLS labels.

Normal packets are classified at the ingress router of the MPLS enabled network. MPLS header is put on the packets depending on the kind of routing and service, we want from the packet. These labeled packets are then used by the LSRs to forward the packets. This is done by looking into the forwarding table which was constructed when the labels were distributed during LSP set-up. This process of forwarding using labels is faster than the forwading done by normal IP routing tables. When an incoming packet reaches the LSR, the incoming label is swapped by the outgoing label and the packet is forwared to the next LSR. This label-switching process is similar to ATM VCI/VPI processing. Inside the MPLS domain, packet classification, forwarding and QoS service are determined using labels and COS. When the packet leaves the domain, the MPLS header is removed.

Summarizing, MPLS provides following ad vatages:

1. Faster routing and processing.
2. Efficient tunneling mechanism.

Both these features make it effective for TE (Traffic Engineering). Details can found in [17] and [18].

MPLS can be used with DiffServ to provide effective QoS as discussed in next section.

1.5.4 DiffServ over MPLS

Though traffic engineering is realized by MPLS, the DiffServ is required for scalable QoS control. The DiffServ over MPLS can map multiple BAs of DiffServ to a single LSP of MPLS. By this, traffic on the LSP can be forwarded based on the PHB of BA. The E-LSP to allow assigning multiple BAs to a single LSP using the EXP field and the L-LSP to allow assigning a single LSP to a single BA (displays multiple packet discarding priorities) can be used for LSP and BA mapping.

E-LSP: The E-LSP shows the PHB of a packet using the EXP field of MPLS shim header. Up to eight BAs can be mapped in the EXP field.

L-LSP:

The L-LSP determines the packet scheduling characteristics based on the MPLS label and the packet discarding priority based on the shim header or layer-2 packet discarding mechanism. The native ATM uses the L-LSP as it cannot use the EXP field.As the Network Elements (NEs) replace packet labels hop by hop, the label and DSCP mapping is difficult to manage. While the E-LSP is easier to control than the L-LSP as it can previously determine the mapping between the EXP field and DSCP of each packet on the entire network.

1.6 Organization

In this chapter, we prepared our background for network technologies (like MPLS, DiffServ) that are used for QoS enforcement. We also defined QoS precisely that would be used in our context.In the next chapter, we formulate our problem statement based on the background that we have prepared in this chapter. Then in the chapters to follow we study in detail the approach to our problem statement. In later sections, we discuss our proposed method, numerical results followed by future work and conclusion.

Chapter 2

Problem Formulation and Related Work

2.1 Introduction

In the present enterprise scenario, there are different applications such as e-mail, Enterprise Resource Application (ERP), video, audio running on the common network infrastructure. All these applications demand different Quality of Service (QoS) from the network, thus making it difficult for enterprise and other crucial applications to be provided differentiated services. Though, the technologies for providing differentiated services like MPLS, DiffServ, IntServ exist, there is a need to translate application QoS requirements to the network configurations. Application level QoS parameters are response time, throughput and availability whereas network QoS parameters are jitter, delay, loss and throughput. This mismatch is to be fixed by providing a mapping to go from application QoS to network performance metrics

[1].In our work, we have focused our attention to enterprise networks and provide such differentiated service levels in MPLS domain, since we have control over the network QoS in the enterprise domain as compared to the Internet. Rest of the chapter is organized as follows. In Section 2.2, we formulate our problem and then in Section 2.3, we discuss existing approaches.

2.2 Problem Formulation

Increasing importance of QoS for applications highlights the need to have a differentiated level of services on the network on which these applications are run. Best-effort networking platforms are not sufficient for this purpose. Thus, applications that require a certain level of QoS must be able to specify their requirements in a clear and accurate manner by using QoS parameters. The values of these parameters reflect the requirements of the application. These parameters can be stored as a pre-defined profile or can be accessed through Application Programming Interface (API). Therefore, for a given application a user using it should be able to invoke an application QoS profile which would be mapped to network configurations and the required end-to-end guarantees would be provided.As an example, suppose an enterprise client has multiple VPN sites across a MPLS network of a Service Provider. Now, the service provider should be able to provide requested QoS to the applications which could be an enterprise application such as ERP, Customer Relationship Management (CRM) or a video-conferencing and provide end-to-end guarantee by reserving resources in MPLS domain. The application QoS profile for video-conferencing should mention high bandwidth, low delay, low delay variation and low packet loss quantitatively or qualitatively in its specification. ERP application QoS profile should mention moderate bandwidth, low latency and low packet loss.MPLS is used for traffic engineering and explicit routing, and therefore, it is complementary to DiffServ technology which provides per-class service differentiation. In the later part of the work, we would explore mapping application QoS to network QoS for applications in DiffServ over MPLS networks i.e. the allocation of bandwidth using bandwidth brokerage architecture in DiffServ along with MPLS explicit path set-up would be explored.There are few applications which work sufficiently well in present networking and operating platforms. But, there are enough applications which need to specify their QoS requirements to the underlying network and receive it. Best-effort networking platforms are not sufficient for them. Thus, applications that require a certain level of QoS must be able to specify their requirements in a clear and accurate manner by using QoS parameters. The values of these parameters reflect the requirements of the application. These parameterscan be stored as a pre-defined user profile or can be obtained through direct interaction with the user.

The achievement of desired QoS is done by resource reservation of the underlying networks and systems which include bandwidth, processing time, memory, hardware etc. Such mechanism to reserve resources at the network level is done by different resource reservation protocolslike RSVP, DiffServ. But, the middleware needs to handle the complexity of mapping the application level specification and requirement to the underlying network QoS specification. These middlewares refereed to as QoS architectures are responsible for providing mechanisms for specification and enforcement of QoS that make use of the resource reservation protocols provided by the underlying system. QoS architectures deal with issues such as the translation of QoS parameters comprehensible at the application-level into the parameters understood by the underlying reservation protocols that control access to the resources provided by the system. Without the services provided by a QoS architecture, these issues would have to be dealt with by the application which would make the programmers as well as users task complex and difficult.

Research into QoS architectures ([5], [6], [9]. [11] and as discussed in later chapters) show folowing features

- Most architectures need a higher level abstraction at the application level. And this what they aim at.
- Most architectures assume a particular underlying network and then build a middleware that provides QoS to the applications above like Aquila (European IST funded project) assumes DiffServ network, QoS-A assumes ATM network etc.
- Usually, the mapping is done in a static manner to some network services created/assumed at the underlying network.
- Resource allocation modules co-ordinate at the middleware in a distributed environment using architectures like CORBA etc.

Also, to most appropriately assign QoS resources network-wide, the technology needs to have a mechanism which calculates and provides the required resources based on the network state and application requests. DiffServ over

Multi Protocol Label Switching (MPLS) is a perfect choice that provides full QoS control over the network. Details of DiffServ over MPLS are explained in the earlier chapter.

Thus, the objective of the thesis is to come up with a mapping mechanism. The core issue in the architecture will be the mapping between SLA at the application level to the QoS parameters at the network level. We will create a standard language to specify application QoS requirements and then be able to map them to the network parameters.

2.3 Related Work

First, we'll talk about diffserv technologies. There has been a great interest in Diffserv architectures both by operators and industry. But, it couldnot achieve large-scale deployment because of missing pieces in DiffServ architecture which are :

1.Absence of proper service definitions provided by the network.

2.Dynamic service creation and automatic configuration management.

3.Traffic Engineering like the way it is done in MPLS.

4.Dynamic Service Invocation.

5.Monitoring

6.Inter-domain QoS aspects.

All above factors are explained in detail in [7]. To deal with above issues, European University has founded 3 projects in the area of QoS support in large IP networks: AQUILA, TEQUILA and CADENUS. These projects try to extend DiffServ architectures to meet above requirements and resolve above mentioned issues.

[8] is a quite interesting work, but there model maps application level APIs to network services which are then mapped to traffic classes. They have assumed the network to be DiffServ and then their middleware acts as an extention to DiffServ to provide service guarantees which DiffServ alone is incapable of providing. There middeware is divide into 3 layers:

1. Resource Control Agent.
2. Resource Admission Control.
3. EAT (End-user Application Toolkit).

EAT is the part that deals with creation of APIs at the application level. The actual mapping of service level requirements to network QoS is done by resource control agent. [9] suggests that work on QoS-driven end-system architecture must be integrated with network configurable QoS services and protocols to meet application-to-application requirements.

In recognition of this, researchers have recently proposed new communication architectures which are broader in scope and cover both network and end-system domains. The [5] does an extensive survey on the state-of-art of the QoS architectures and mentions following projects:

- Extended Integrated Reference Model (XRM), which is being developed at Columbia University;
- OSI QoS Framework, which is being developed by the ISO SC21 QoS Working Group;

- OMEGA Architecture, which is being developed at the University of Pennsylvania;
- Heidelberg QoS Model, which is being developed at IBMs European Networking Center;
- Tenet Architecture, which is being developed at the University of California at Berkeley
- End System QoS Framework, which is being developed at Washington University;
- IETF QoS Manager (QM), which is being developed by the int-serv group as part of its strategy for an integrated services Internet;
- TINA QoS Framework, which is being developed by the TINA Consortium; and
- MASI End-to-End Architecture, which is being developed at Universit Pierre et Marie Curie.

We now discuss Aquila architecture in detail here.

2.3.1 Aquila Architecture

The general AQUILA network architecture is based on the DiffServ network concept. The objective of the project is to provide dynamic control to QoS based traffic. This is done by creating an architecture with a layerresource control layer for controlling network resources above the DiffServ network. It maintains the distribution of network resources between different network entities, especially the network access points. So it can be assured, that the underlying IP network can provide the assumed Quality of Service for the specific network services.On top of this technology, admission control and resource management mechanisms are built Dynamic adaptation of resource allocation to user requests enables the overall architecture to scale to very large networks.

Resource Control Layer(RCL)

The RCL is responsible for the management of QoS resources inside the AQUILA network. This can be split in three main tasks and modules:

- End-user Application Toolkit (EAT):A graphical user interface is offered to the end-user directly or to the applications run by the user. Using this, any application can request certain resources from the network to run with a specified QoS. It is the interface to the AQUILA QoS infrastructure.

- Admission Control Agent (ACA)Performs policy and admission control at the network edges. Each edge router or border router is controlled by an ACA. Each ACA gets a certain amount of resources by the RCA enabling the ACA to perform admission control autonomously. The ACAs receive requests for new IP flows with specific QoS requirements. Their task is to authorize the requests and to check, if the network is able to support the new flow. For this task, they closely interact with the second main entity, the so-called Resource Control Agents.

- Resource Control Agent (RCA)Monitors and controls the available resources in the network. It is the ultimate authority for the resource handling in the AQUILA network. Based on the prior history of resource usage and actual requests, the RCAs distribute resource shares to the Admission Control Agents.

End-users access the Resource Control Layer by using the End-user Application Toolkit. The EAT does not constitute a new signalling protocol for IP networks. Instead, it can be described as a QoS middleware that brings the functionality of the Resource Control Agents Network into the end-user terminals and servers. The internal signalling protocol used between user terminals and the main network can be based on existing schemes (e.g. RSVP) or even on CORBA or DCOM interfaces depending on application needs. In any case, the Edge Device (ED) analysis the user request and executes the user policy control and the local admission control operations in order to determine whether the specific user has the administrative rights and whether there are enough internal resources for the handling of the particular request. However, end-to-end guaranties can be provided only if the level of available resources of all intermediate routers until the final destination is known. Therefore, the ED uses its interface with the Resource Control Agents Network for performing the network admission control operation.

2.3.2 Problem Approach

We aim to provide the mapping mechanism for SLA to QoS parameters mapping for DiffServ over MPLS networks.We discuss the ideas of Aquila mapping and architecture in the next section as is important in our context, and analyze their approach.And then elaborate an approach and strategy for our work in the next chapter.

sectionSLA to QoS Mapping

2.3.3 SLA to QoS mapping scheme

The paper [3] by Hyo-Jin Lee, Korea mentions a mapping mechanism between QoS parameters (here, it means application level QoS parameters) and Network Performance Metrics (NPM).Here, these people have given an architecture of a SLA monitoring system and have given a method to map QoS to NPM. They areexperimenting with their mapping function by applying it to IP backbone network.

A SLA usually mentions the following QoS parameters: Delay, latency, delivery and availability. Now, given above parameters we should be able to map them to network performance metrics. We should also be able to influence and use this information of SLA breaking for billing, reporting and even, use as a feedback for the network to restore its status such that the network again conforms to the SLA.This kind of system is known as Service Level Management System. It performs 2 functions: SLA monitoring and SLA provisioningThe architecture of the same is given in the paper mentioned above.Here, we collect NPMs from the network through SLA agent and then map them to QoS. Say, some application wants a delay of less than 10 sec. We should be able to collect the Network state (which is nothing ,but NPMs which are: traffic congestion, throughput, availability and latency).Above mapping as done in paper is shown for network access services like IP-VPN, leased-line service, xDSL. It is yet to be extended to application and content services like web-hosting etc. But, this scheme is quite generic and is not precise. It can only serve as a guideline to serve a particular need or a particular situation wherein the entire mapping mechanism would need to be reworked.

2.3.4 QoS Specification Languages

Most research in the QoS specification deal only with QoS specification at the application level and had little to do with the SLA to QoS mapping.People have come up with languages with QML, HQML, CQML etc which specifies QoS at the application level and then the applications above interact to ensure that the QoS is met or not. The following example tries to make this clear.

Say, a client which has its QoS requirements specified in QML requests for a particular service from a server which may or may not be capable of providing it the required service. Thus, they interact say in the CORBA environment and fulfilll the request. These have little to do with the co-relation with the network performance metrics.

[4] comes with a language SLAng, a language for SLA (service level agreements) which describes the proper syntax for SLA. These people say that SLA can be of 2 kinds : vertical and horizontal. Horizontal SLAs are the ones at one of the following levels:1. Application level.2. Container level.3. Network level.And then there are vertical SLAs which maps the SLAs vertically which clearly is also our objective. But, he just came with the way to represent SLAs starting from SLS (Service level specification) of the Tequila engineering work.

2.3.5 Aquila Mapping Scheme

The application level services which are specified by the user through a browser specifying a application level profile in XML created by Aquila group. This partial information is then mapped to servicecomponent profile (again in XML) which are detailed requirements of the resources that we need to reserve from the network.

The mapping is static meaning the classification of network services as is done above in above 5 categories is sufficient to meet almost all application level requirements. And for any given application requirement, we map it to one of these services that most closely meet its requirement.

Before committing a resource reservation request, some admission control algorithm is used that ensures if sufficient resources are available to meet the request. Above, method assumes DiffServ network. They have beautifully and effectively solved the problem of mapping SLA which may be specified using QML, HQML or using Aquila's applicationprofile DTDs and service component DTDs into SLS (technical aspects of SLA requirements of applications) and then into 5 network services that most closely meet its requirements.

The 5 network services as done in Aquila architecture are:

1. Premium Constant Bit rate (PCBR).
2. Premium Variable Bit rate (PVBR).
3. Premium Multimedia (PMM)
4. Premium Mission Critical (PMC)

5. Standard Best Effort (STD).

The 5 network services are provided by creating 5 traffic classes (TCL) as follows:

- First TCL has a queue and has the highest priority.
- Rest three are respectively in smaller priorities and controlled using weighted fair scheduling.
- Last one is of the smallest priority and is best-effort service. Yet, some resources are reserved for it too.
- Different network parameters such as bandwidth, peak rate, average rate are controlled using leaky token bucket (and sometimes, dual leaky token bucket to control average traffic rate for a period of time, also).

The above details of traffic classes are then used to allocate resources in DiffServ network.

The end to end state is ensured by maintaining the logical resource pool in the layer above network that keeps track of the resources. This logical layer is known as RCL (resource control layer).

Thus, to implement above mapping schema, to MPLS networks, we would need to map network services to the MPLS TE. Thus, bringing changes to the RCL (resource control layer).

Aquila Architecture has already been explained in detail in the earlier section of related works.

2.3.6 Extending Aquila : SLA to QoS mapping for DiffServ over MPLS network

In order to be able to provide proper SLA to QoS mapping in MPLS networks, we need to create network services that could be mapped to the service component profiles of the applications. These network services can be crested by considering following issues which would make it a traffic engineering problem:

1. We need to be able to provide 5 network services as is done for above DiffServ network.
2. We need to create a servicecomponent network level profile that could be mapped to the application profile which are the application/user specified requirement.
3. In MPLS networks, we need to map LSPs and labels and see how they can be used to provide 5 network services as mentioned above.

Chapter 3

Problem Approach

We have named our architecture as Q-MPLS. The goals of the Q-MPLS are:
(1) To come up with application profiles that rightly specify QoS requirements of different applications.
(2) To map application level requirements to the network level details i.e. MPLS level details.
The approach involves the creation of few application services which would meet the need of most applications. These applications services are offered as products to the clients and the client can chose to select any one application service profile for his application (that best suits his applications QoS requirements). These application service profiles are nothing, but SLA offerings to the clients. This SLA or application service profile can have following parameters: delay, jitter, throughput, packet-loss and bandwidth. There will be few application profiles that would cover most of the applications requirements. Aquila architecture has incorporated 5 such application service profiles which are premium constant bit rate (PCBR), premium vari-

able bit rate (PVBR), premium mission critical (PMC), premium multimedia (PMM) and best-effort drawing direct relation from the ATM classes. If the customer does not specify the application service that he intends to avail, then the mapping of the application to the appropriate network service is done automatically at the ingress router by looking at the application QoS requirements and traffic type.

The customer chooses one such service. These application service profiles are mapped to network services. Network services are network level MPLS over DiffServ specifications. The mapping is done by admission control agent known as QMA (QoS MPLS Admission agent). QMA checks whether the request for the new service can be granted. If there are available resources, it allocates the resources. The QMA uses ping-probe method or measurement based admission control to perform admission test. Ping-probe algorithm is explained in next section. QMA is present at the service provider network edge as shown in figure 1. QMAs may communicate among themselves or not depending on the kind of admission control algorithm used.

The QMA does application mapping to network service (if it is not done by the application user), performs calculation and generates configurations. These configurations are finally used to set Label Switched Path that provide the required network service. Finally, the measurement architecture measures the performance of the application and checks that that application QoS is indeed met.

There is one-to-one mapping of application services to network services (which is also known as class type), but it can be otherwise as well. One-to-one mapping simplifies the design. We chose to use 5 application service

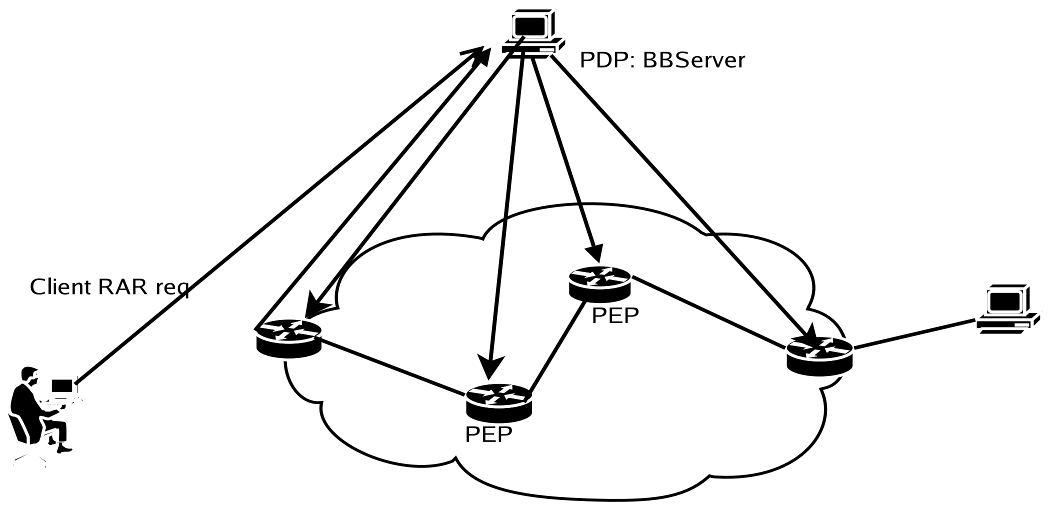

Figure 3.1: Q-MPLS architecture

profiles including best-effort traffic analogous to 5 application service classes of Aquila architecture. The 5 application service profiles would correspond one-to-one (which can be otherwise also) with 5 network service profiles or class-type profiles which are explained in detail in next section. These 5 network services are MPLS over DiffServ implementation of each application service profile.

3.0.7 Application Service Classes

There are 5 application services that are envisaged which are offered as service offerings to the customer. The customer choses one such profile and requests QoS for his application. This application service request is then enforced on the network. The 5 application service classes are shown below:

Application Service class	Applications	examples
0	Real-time, jitter sensitive constant bit rate	voice
1	Real-time variable bit rate	video
2	Greedy TCP	FTP
3	Non-greedy TCP	database queries
4	Traditional IP applications	

Thus, a customer chose one of the above application profile depending on the kind of application he intends to run over the network. The application profile consists of 2 components: Application user specification and traffic characteristics specification.

Application User Specification

Application user specification component specifies application in its raw form along with the application service it intends to avail. The specification can be in XML or otherwise. XML specification for Winamp audio application would be:

$< ApplicationUserSpecification >$

$< name > Winamp < /name >$

$< traffic > audio < /traffic >$

$< transportcharacteristicstype = UDPport = 2030/ >$

$< applicationserviceclass > 0 < /applicationserviceclass >$

$< qosrequest >$

$< delay > 100ms < /delay >$

$< jitter > 40ms < /jitter >$

$< /qosrequest >$

$< /ApplicationUserSpecification >$

Other example application profiles are:

Profile 1:

Application: VLC player.

Traffic : Audio, Video

Transport Characteristics: UDP, 2040

Application Service class: 0

QoS request: Nil.

Profile 2:

Application: VLC player.

Traffic : Audio, Video

Transport Characteristics: UDP, 2040

Application Service class : 2

QoS request: Nil.

Profile 3:

Application: VLC player.

Traffic : Audio, Video

Transport Characteristics: UDP, 2040

Application Service class: 5

QoS request: Nil.

Given, above application service profiles we should be able to enforce it on the network. QoS request parameters, if mentioned can be used to determine if for a chosen application service class, the QoS requested can indeed be met or not. Example, for a service class 0, the end-to-end delay should be less than 100ms and if the QoS requested reflects application service class as 0 and end-to-end delay as 200ms, error is returned to the user. All the applications in a given application service class receive similar treatment.

Traffic Characteristics Specification

Along with above mentioned application service profiles, customer also needs to specify the kind of traffic that he is injecting or intends to inject into the network. This information is used for admission control purposes and settling customer's contract with service provider. This information can also help in shaping the incoming traffic. This information is represented as a service contract which is a tuple as follows: (Source IP address, sink IP address, Traffic characteristics).

Traffic characteristics can have following components:

$/*flowspecificationidentification*/$
flow-id;
$/*video, voiceordata*/$
media-type;
$/*framesize*/$
frame-size;
$/*tokengenerationrate*/$
frame-rate;

$/ * size of the burst * /$

burst;

$/ * max transmission rate * /$

peak-rate;

$/ * bandwidth requirement of the application * /$

bandwidth $/ * time for which flow is expected to exist * /$

flow-time; Here, throughput is described in terms of frame size, frame rate, burst size and peak rate. The frame size and frame rate represent the average throughput requirement, while the maximum throughput is represented by the peak rate performance parameter. In addition, the flow spec gives the potential burstiness of the incoming traffic using the burst parameter. Flow-time is the time for which the flow would exist in the network and therefore, can be used for admission control determination for future flows. Thus, this parameter can be incorporated in admission control algorithms. Often, it is difficult for most customers to know the traffic characteristics of their applications, this information can be cached at the service provider for most common applications and can be used by the customer. And to avoid customer making misuse of this information, shaping of incoming traffic can be done at the ingress routers. This would ensure that the customer indeed is sending the traffic that he has approved or specified in his traffic characteristics profile. Example XML traffic characteristics profile for Winamp audio application:

<TrafficCharacteristicsSpecification name=Winamp

flow-id=1 media-type=audio expected-flow-time=1hr">

<sourceIP> 10.129.76.43 < /sourceIP>

$< sinkIP > 10.129.76.83 < /sinkIP >$

$< trafficcharacteristic >$

$< frame - size > 550 bytes < /frame - size >$

$< frame - rate > 64 kbps < /frame - rate >$

$< burst > 80 kbps < /burst >$

$< peak - rate > 67.5 kbps < /peak - rate >$

$< bandwidth > 64 kbps < /bandwidth >$

$< /trafficcharacteristic >$

$< /TrafficCharacteristicsSpecification >$

Above profile information helps in keeping the incoming traffic in shape and thus, keeps track of the resources used. Thus, for the Winamp audio application the entire application profile would be:

$< ApplicationProfilename = Winamp >$

$< ApplicationUserSpecification >$

$< name > Winamp < /name >$

$< traffic > audio < /traffic >$

$< transportcharacteristicstype = UDPport = 2030/ >$

$< applicationserviceclass > 0 < /applicationserviceclass >$

$< qosrequest >$

$< delay > 100 ms < /delay >$

$< jitter > 40 ms < /jitter >$

$< /qosrequest >$

$< /ApplicationUserSpecification >$

$< TrafficCharacteristicsSpecificationname = Winampflow - id = 1$

$media - type = audio$

$expected - flow - time = 1hr >$

$< sourceIP > 10.129.76.43 < /sourceIP >$

$< sinkIP > 10.129.76.83 < /sinkIP >$

$< trafficcharacteristic >$

$< frame - size > 550bytes < /frame - size >$

$< frame - rate > 64kbps < /frame - rate >$

$< burst > 80kbps < /burst >$

$< peak - rate > 67.5kbps < /peak - rate >$

$< bandwidth > 64kbps < /bandwidth >$

$< /trafficcharacteristic >$

$< /TrafficCharacteristicsSpecification >$

$< /ApplicationProfile >$

3.0.8 Network Service Classes and their Traffic Engineering for DiffServ over MPLS network

Network services in MPLS over DiffServ network are defined in terms of CTs (class types). In [19], few MPLS-DiffServ-TE class types (CTs) are defined. Each application flow corresponds to some class type (network service class) and the parameters in the network are configured and propagated per-class-type. Thus, each class-type is an aggregation of individual classes that can receive similar treatment of performance levels. The parameters are configured and propagated per-class-types (CTs) and not per class on each LSR interface. However, no bandwidth requirements are enforced for classes within a CT. As explained in [19], CT is a set of classes that satisfy the following two conditions:

1. Classes in the same CT have common aggregate maximum bandwidth requirements (and, if applicable, common minimum bandwidth requirements) to satisfy required performance levels.

2. There is no maximum or minimum bandwidth requirement to be enforced at the level of individual class in the CT.

We can consider different generalization of CTs and as a combination of:

1. DiffServ/queuing priority PHBs,

2. QoS classes (e.g., as specified in Draft Recommendation Y.1541 [20],

3. MPLS/CAC priority classes,

4. Restoration priority classes at both the MPLS-LSP and transport link level, and

5. Preemption priority classes.

We consider 6 such class-types which are enforced as follows:
Class type 0: DiffServ EF PHB; MPLS CAC and restoration: High (for control traffic)
Class Type 1: DiffServ EF PHB; MPLS CAC and restoration: normal.
Class Type 2: DiffServ AF1x PHB; MPLS CAC and restoration: high; LSP preemption allowed.
Class Type 3: DiffServ AF2y PHB ; MPLS CAC and restoration: normal ;

LSP preemption not allowed;
Class Type 4: DiffServ AF3y PHB ; MPLS CAC and restoration: low; Any route/path.
Class Type 5: DiffServ best-effort PHB; MPLS CAC and restoration: best-effort. Separate Queue. Any route/path.

For class type 0 and 1, we have separate queue with preferential servicing and traffic grooming. Constrained-based routing (CBR) and distance is also considered. CT2 has lesser constraints on CBR and distance than CT1. There is a single queue for CT 0 and 1. For class type 2, we have a separate queue that is used for variable bit rate traffic. For class type 3 and 4, separate queues with drop priority are considered. Class-type 4 has lesser constraint based routing and distance constraint than CT 3. CT 5 has a separate queue to give minimum bandwidth to best-effort traffic and has any route/path implementation.Above class-types (also referred to as network service classes) are mapped one-to-one with application service profiles leaving CT 1 for the control traffic. Thus, application profile 0 would map to class-type 1, application service profile 1 to class-type 2 and so on.

3.0.9 QoS Architecture based on above CT classifications:

The above class-types and their mapping to application profiles of table 1 gives us a clear mapping for building of a proper framework for SLA to network configuration. As discussed earlier, our objective is to map the above specification to one of the CT. Thus, if we know the kind of application, traffic characteristics and the QoS based on it as follows, we can come with the exact CT and thus, provide end-to-end guarantee:Example:

Application Service Profile:

Application: Winamp

Kind: Voice

Traffic-kind: real-time and jitter sensitive.

Application service profile: 1

E2E delay: <= 100ms

delay variation <= 50ms.

loss ratio <= $10^{-}3$.

Traffic characteristics:

Source IP: 10.129.76.43

Sink IP: 10.129.76.83

flow-id : 1

media-type: Audio

frame-size: 550 bytes

frame-rate: 64kbps

burst: 80 kbps

peak-rate: 67.5 kbps

bandwidth: 64kbps

flow-time: 1 hr

We can know the CT which will be 1 as application service profile is 0. Its implementation is done as follows:Per hop DiffServ behavior: EF. Separate queue with higher preference and preferential servicing, traffic grooming.

There is single queue for CT 0 and 1.Higher LSP and admission control priority for CT 0 then 1. Burst size, frame-size, frame-rate and peak rate, since, mentioned would help in keeping the incoming traffic profile in shape, and thus, would help keep track of the resources used. Thus, an admission control agent (ACA) can be built. ACA keeps track of the resources and thus, perform calculations of the resources consumed. ACA uses this information to perform admission test on the flow requesting admission. A flow is admitted only when it meets its QoS requirements. This kind of admission control is known as Measurement-based admission control (MBAC). Other alternative for the admission control that can be explored is probe-based method. Probe-based admission control (PBAC) is mentioned in detail in [21] and [22]. Detailed description of the QoS architecture for the same has been dealt in the next chapter and has been named as Q-MPLS.

Chapter 4

QMPLS Implementation

4.1 Introduction

Our architecture which we have named as Q-MPLS has following goals:

1. To come up with a specification language in XML that allows application to request QoS from the network.

2. To map these application specifications to network level configurations for DiffServ over MPLS network.

Q-MPLS has 3 components as follows:

1. End-user QoS specification Language Component.

2. DiffServ over MPLS Admission Agent

3. Network Resource Agent.

2 and 3 functions above are together performed by QMA (QoS MPLS Admission Agent). The approach involves the creation of few application services which would meet the need of most applications. These applications services are offered as products to the clients and the client can chose to select any one application service profile for his application (that best suits his applications QoS requirements). These application service profiles are nothing, but SLA offerings to the clients. This SLA or application service profile can have following parameters: delay, jitter, throughput, and bandwidth. There will be few application profiles that would cover most of the applications requirements. They has used 5 application service profiles which are premium constant bit rate (PCBR), premium variable bit rate (PVBR), premium mission critical (PMC), premium multimedia (PMM) and best-effort drawing direct relation from the ATM classes. These application QoS are specified in XML.The customer chooses one such service. These application service profiles are mapped to network services. Network services are network level DiffServ over MPLS specifications. The mapping is done by admission control agent known as QMA (QoS MPLS Admission agent). QMA checks whether the request for the new service can be granted. If there are available resources, it allocates the resources by converting to network configurations. The QMA has a measurement component that measures and maintains network resource state, performs calculation and generates configurations. These configurations are finally used to set Label Switched Paths (LSPs) that provide the required network service. The database maintained at the QMA at the edge-router keeps track of the entire network services explicit path and other control parameters such that for each network ser-

vice, the QoS offerings are met by interacting with other QMA agents in the network. Figure 4.1 shows the architecture of Q-MPLS. There is one-to-one

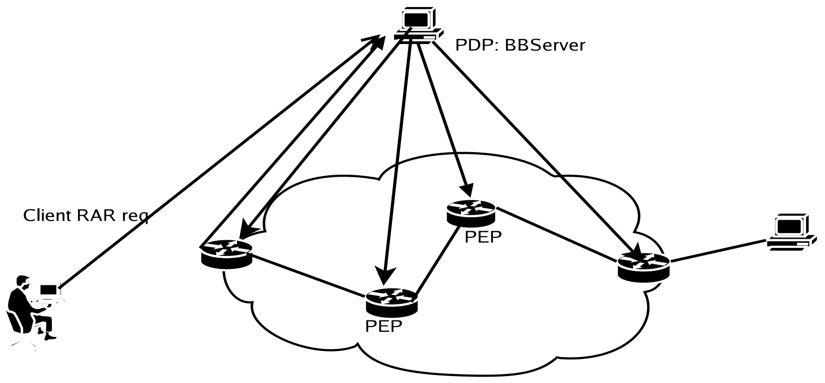

Figure 4.1: Overview of Q-MPLS

mapping of application services to network services, but it can be otherwise as well. One-to-one mapping simplifies the design. A Sample network service profile is given below:

$< TechnicalCharacteristic >$

$< Signallingtype = "RSVP - TE" routingAlgorithmType = "CSPF" >$

$< LSPathingress = 10.3.2.2.1 outgress = 10.105.1.11 >$

$< Node >$ 10.3.2.21 $< /Node >$

.....

$< Node >$ 10.105.1.10 $< /Node >$

$< /LSPath >$

$< OptionsdeploymentType = IntServ >$

....

$< /Options >$

$< /Signalling >$

$< /TechnicalCharacteristic >$

Above network service profile gives the signaling type or Label Distribution Protocol (LDP) used which can be RSVP-TE or CR-LDP along with the kind of routing algorithm the network is using. Then, LSPath mentions ingress and egress node Internet Protocol (IP) addresses followed by the IP address of the intermediate nodes of the path taken by LSP. The measurement architecture should use different tools to maintain the state of the network and maintain the active collection of network metrics to assist in the auditing of application QoS profile for the service.

4.1.1 QMPLS Implementation

QMPLS implementation is done using COPS (Common Open Policy Service) protocol. We have a BB server also called a bandwidth or resource broker. When a flow request comes, the request (known as RAR, Request Allocation Request) is communicated to the BB Server. BB Server (bandwidth broker or BB) which acts a COPS server (also known as PDP, policy decision point) decides whether to permit the flow request or not. If the flow request is accepted, the decision is communicated to Linux routers (also known as PEP, Policy enforcement points). The protocol used between PDP and PEPs is COPS-PR for which we have used an implementation java package from New South Wales University. For each domain, we have one PDP.We are working on only one domain and therefore, we have only one instance of PDP working. PDP keeps track of all the resources. All PEPs are connected to PDP and are all the ingress-egress routers where configurations for an incoming traffic flow are enforced.

A customer signs up for a particular SLA which includes: starting date, starting time, end date, end time, service class (PCBR, PVBR, PMM, PMC, BE), total bandwidth request. This information is stored in the database and the administrator allocates him the SLA as per the management or administrative policies. Currently, no validation on the correctness of SLA is done in the program. When a flow request comes to the ingress or the QMA (admission control agent), it contacts BB Server which keeps track of all the resources in the network. The client is then asked to input the QoS profile for the input traffic (which is stored as .prf file. There are different QoS profiles stored as Video.prf, FTP.prf, CBR.prf. These are common QoS profiles of common applications that run on the network and represent the kind of QoS that the application expects from the network. There could be custom application QoS profiles.After the QoS profile, service class (PCBR, PVBR, PMM, PMC, BE) is entered. This service class for the application should be same as the SLA agreement of the customer. Then the starting date, starting time, end date, end time, bandwidth request are entered for the flow. Then the admission control algorithm as discussed later is executed by the QMA in co ordination with BB Server.If the result of admission control is positive, flow is admitted by enforcing configurations on PEPs (ingress and egress routers). The configurations enforced include traffic policing, marking of incoming traffic for particular traffic class and set-up of LSP. PEP is a Linux DiffServ over MPLS ingress, core or egress router. When the PEP, first starts or is run, it connects to BB Server and executes the default initialization configurations which are:

- Default queuing set-up: i.e. DiffServ Queue setup of EF, AF and BE.
- Running the default RSVP daemon along at each node which would permit the setting of LSP when the flow would be admitted.

When a flow request is to be admitted, PDP commands PEP to execute following during the flow request:

- Marking, Metering, Policing, shaping and scheduling, of the flows at each node (depending on whether it is ingress or egress) using the DiffServ class (EF, AF or BE) it belongs to.
- Setting up the LSP between the ingress and the egress i.e. between source and destination along the least cost path.
- If some other flows of lower service classes are to be preempted, then preempting them.

Now a user requests a SLA to the PDP (or BB) which is stored in the database. Now when a user requests a RAR i.e. makes a QoS request, admission control agent (QMA) in co-ordination with BB Server:

1. Decides on the DiffServ class of the flow request.
2. Commands the PEP (ingress) to execute proper policing and filtering mechanism to control the traffic injected into the network. Application-based policing and filtering can be done based on port, source IP and destination IP address.
3. Decide on the path to be taken in case shortest path is not reachable. For this different algorithms can be worked out. Use open-source totem.

4. Finally set up the LSP along a path that meets the requirement with the DiffServ class as decided upon in 1.

Now, we need to ensure and check that the QoS is indeed met. For this, we use D-ITG (distributed internet traffic generator [25]).

4.1.2 Scheduling and Traffic Class Implementation

As seen in Aquila Architecture, it is advised to have separate service classes for CBR and VBR kind of UDP traffic and different service classes for TCP greedy and short-lived traffic type. Therefore, each service class would have different queuing, policing, traffic specification and admission control mechanism for DiffServ over MPLS network as shown below:

The service of traffic class in this context is defined as the preference that the traffic class gets over other traffic classes to do the followings:

1. *Order of preference of scheduling over other traffic class*

2. *Order of path-rerouting and preemption in case higher sensitive traffic class needs to send the data along the path that is occupied by lower traffic class*

Scheduling mechanism used in our case at the routers is PQ (Priority Queue). Thus, observing that we need to give CBR traffic preference over all other traffic classes, we give higher priority to PCBR traffic. For VBR UDP traffic, we need to give next highest preference followed by the remaining Traffic Classes (TCs). This can be done using PQ to PCBR, PVBR and other remaining classes. We therefore, use high PQ for TC1, medium priority for TCL2 and lower priority for remaining TCs.

Since, we use PQ which can starve lower TCs, our AC should take care that lower TCP TCs are not starved. Thus, we need to distribute bandwidth among each TC such that the QoS requirement of each TC is best met and the network resources are also most utilized. We also need to calculate the buffer size that would prevent packet loss and also keep the packet delay across the network within bounds.

4.1.3 Complete Admission Control Algorithm

Algorithm 1

Finding Critical Links using MIRA (Minimum Interference Routing Algorithm):

We have incorporated MIRA as proposed in [26] in our admission control algorithm. It takes into account ingress and egress locations and attempts to route the LSP along the path which offers minimum interference with the possible future requests. Specifically, an incoming connection request between (S_i, T_i) is routed with the goal of maximizing an objective function which is either the minimum maximum-flow (maxflow) of all other ingress-egress pairs or a weighted sum of maxflows, where weights α_{ST} assigned to each ST pair reflect the "importance" of the flow. Let the network topology be defines as a matrix. When a new call has to be routed between the source/destination pair (S_i, T_i), MIRA determines the set L_{ST} of the critical links for all the source/destination pairs (S_j, T_j) than (S_i, T_i). The weight w of each linkl is then set according to the equation w(l)= $\Sigma_{(S,T):l\epsilon L_{ST}} \alpha_S T$, and the route which causes the minimum interference to other source/destination pairs is selected. For our implementation, we have taken, α_{ST} as

1. Initially, we have taken the weight of each link as 1. Therefore, the new flow is routed along the path with least number of critical links.

2. As the new flows are admitted, the flow importance is added to the ingress-egress link weight along the path between ingress and the egress. This decreases the probability of new flows to be admitted along the same path or it admits new flows such that the common links are avoided.

Admission Control

- For PCBR:

  - Flow specification:

    Implicit QoS guarantees: packet-loss (almost zero)<=$10^{-}8$, jitter 0, delay $<$ 150ms. In-profile hard QoS, out-profile packets are dropped. no packet misordering.

    Explicit QoS Spec:
    Class Type: PCBR
    QoS specs: PR, m (PR is peak rate and m is minimum policed size)
    Weight/Importance of the flow: 1,2 or 3.
    preemption (yes/no): default is NO.

Explicit QoS specs should check that the QoS requested respects implicit QoS that the class guarantees.

NOTE: weight should reflect weight and more so, bandwidth/resource usage. Thus, giving weight can be automated such that more are the resources consumed, more is the weight of the flow. But, for simplicity we assume that the user provides the weight to the Admission control agent.

– AC Algorithm

1. Find a path with minimum weight due to CLs along its path.
2. Admit flow if:

   $PR_{new} + \sum_k PR_k <= \rho r_i$
   as
   where
   $\rho = \frac{2B}{2B - lnP_{loss}}$ gives the admissible load such that the maximum packet loss, P_{loss} (which is 10^-8 in our case of CBR class) is ensured.
3. If not then repeat 1 else admit the flow.
4. If no feasible path is found, then look for flows that are lower in weight (importance) and have preemption allowed. NOTES: 1.We have assumed that the delay along all paths is same. We can consider the difference in our algorithm and route the flow along path that meets the delay requirement

even within a particular (say, PCBR) class.

  - Administrator Settings:

    1. Single token bucket meter and dropper with r = PR and b (bucket size) = x * M. x= 2 keeps jitter low. x can be between 1 and 5.
    2. Buffer size of the routers is calculated as explained above.
    3. Codepoint: 110—000—xx.

  - Example Application: voice application (compressed or uncompressed)

- For PVBR:

  - Flow specs:
    Implicit QoS guarantees: packet-loss <=10^-4, delay <=150ms. In-profile packets are provided looser QoS than PCBR, out-profile packets are dropped. no packet misordering.

    Explicit QoS Spec:
    Class Type: PVBR
    QoS specs: SR, BSS, PR. SR is sustainable rate, BSS is Bucket size for sustainable rate, PR is peak ratpreemption
    Weight/Importance of the flow: 1,2 or 3.
    preemption (yes/no): default is NO.

    Explicit QoS specs should check that the QoS requested respects implicit QoS that the class guarantees.

- Algorithm:
  1. Find a path with minimum weight due to CLs along its path.
  2. Admit flow if:
     $Eff_{new} + \sum_k Eff_k <= r$
     where Eff is defined in [28] and r is the admission control limit for this class. P_{loss} is 10^4 in our case of PVBR class.
  3. If not then repeat 1 else admit the flow.
  4. If no feasible path is found, then look for flows that are lower in weight (importance) and have preemption allowed.
     NOTE: 1. We'll have to keep part of the PVBR traffic bandwidth for Control and OSPF traffic also.
- Administrator Settings:
  1. Dual token bucket meter and dropper with r1 = SR and b1 (bucket size) = BSS, r2=PR and b2=x2 * M. x2= 2 keeps jitter low. x can be possibly between 1 and 5.
  2. Buffer size of the routers scheduling as follows: d= h* n_2 * M / R where R is scheduling rate and n_2 is buffer size and h is the number of hops between ingress-egress. We take maximum possible number of hops.
  3. We can calculate N=number of homogeneous flows. This would give us row * r_i i.e bandwidth permissible for VBR flows. r_i is the maximum AC limit at ingress i.

here $D = r_i/PR_{min}$, $P_{loss} <= 10^-4$, B is calculated above.

4. Code point:= 101—000—xx

– Example Application: live video transmission.

- For PMM:

  – Flow specs:

  Implicit QoS guarantees: TCP traffic, packet-loss <= 10^-3, no QoS for out of profile packets. no packet misordering.

  Explicit QoS Spec:

  Class Type: PMM

  QoS specs:preemption

  Weight/Importance of the flow: 1,2 or 3.

  preemption (yes/no): default is NO.

  Explicit QoS specs should check that the QoS requested respects implicit QoS that the class guarantees.

  – Algorithm:

  1. Find a path with minimum weight due to CLs along its path.
  2. Admit flow if: $SR_{new} + \sum_k SR_k <= r_i$ and $N_3 * M_3 <$ B where SR_{new} is the sustainable rate of new flow, the r_i is AC limit, N_3 is the number flows, M_3 is the maximum packet size, B is the buffer size.
  3. If not then repeat 1 else admit the flow.

    4. If no feasible path is found, then look for flows that are lower in weight (importance) and have preemption allowed.

  - Administrator Settings:

    1. Single token bucket meter and marker with r1 = SR and b1 (bucket size) = BSS
    2. WRED parameters (maxth, minth, maxp) for the flow are set using derivations explained in [28]. Trade-off has to be made between stability (oscillations) and queuing delay.
    3. For buffer size of calculation, we can refer [28] which details the calculation of buffer size for TCP flows for proper QoS enforcement.
    4. Code point:= 100—000—xx (for in-profile packets), 011—000—xx (for out-profile packets).

  - Example Application: streaming video/audio, FTP.

Assumptions for AC: Two essential assumptions for a definition of effective bandwidth for AC purposes are the following:
The effective bandwidth of a traffic stream is independent of the other streams with which it is mixed. The sum of the effective bandwidths of two independent traffic streams is equal to the effective bandwidth of their superposition (the additive property).

- For PMC:

  - Flow specs: Implicit QoS guarantees: TCP traffic, packet-loss <= $10^{-}3$, no QoS for out of profile packets. no packet misordering.

Explicit QoS Spec:

Class Type: PMM

QoS specs: SR

Weight/Importance of the flow: 1,2 or 3.

preemption (yes/no): default is NO.

Explicit QoS specs should check that the QoS requested respects implicit QoS that the class guarantees.

– Algorithm:

  1. Find a path with minimum weight due to CLs along its path.
  2. Admit flow if:
     $Eff_{new} + \sum_k Eff$k<= r as
     $Eff() = maxSR, \frac{PR*T}{B/R+T}$
     where T= MBS/PR, R is the scheduling rate for TCL4 in ingress link, B is buffer size dedicated to PMC class.
  3. If not then repeat 1 else admit the flow.
  4. If no feasible path is found, then look for flows that are lower in weight (importance) and have preemption allowed.

– Administrator Settings:

  1. Dual token bucket meter and marker with r1 = SR and b1 (bucket size) = BSS, r2=PR, b2= x * M. x can take values from [1,5]. Possible value is 2.
  2. WRED parameters (maxth, minth, maxp) for the flow are set

using definitions explained in [28]. Trade-off has to be made between stability (oscillations) and queuing delay.

3. For buffer size of the routers, we can refer [28] which details the calculation of buffer size for TCP flows for proper QoS enforcement.
4. Code point:= 010—000—xx(for inprofile packets), 001—000—xx (for out-profile packets)

– Example Application:database queries.

NOTES: The proposed admission control belongs to the methods based on the effective bandwidth notion assuming RSM multiplexing scheme. In this method the traffic is characterized by three parameters (SR, PR, MBS), where MBS is the maximum burst size submitted with the PR rate (MBS=BSS*PR/(PR-SR)).

Chapter 5

Validation

5.1 Strategy

In order to validate our proposed architecture and algorithm, we perform tests:

- Validation of isolation of classes such that the QoS experienced by flow in one class is not affected by the traffic flows due to other classes. This means that when the traffic rate of some flow is increased in the network, the QoS of flows already in the network are not affected or less affected for DiffServ over MPLS network than over just DiffServ or IP network.

- Effectiveness of admission control and network provisioning. We have to validate the effectiveness of our algorithm by testing that only those flows whose QoS can be met are admitted otherwise rejected. And also to validate that if the flow is rejected then that is the right admission decision taken by the algorithm.

We have performed tests to validate isolation of classes and efficiency of network provisioned classes as discusses in next section.

5.2 Test Network

We implemented QMPLS on a 3 node network test-bed using DiffServ over MPLS Linux machines. To implement, DiffServ over MPLS functionality to the linux kernel (redhat linux-9 kernel), patches from [27] were applied to the kernel 2.4.19. For detailed installation procedure of adding the DiffServ over MPLS functionality over linux kernel, refer [27]. The Diffserv over MPLS patched kernel implements the RSVP-TE daemon to set LSP (Label Switch Path) and has following features ([27]):

- LSP set-up

  Constraint routed LSP set up

  DiffServ over MPLS LSP set up (E-LSP and L-LSP)

  LSP with Intserv style reservations.

- Mapping traffic to LSPs based on:

  Protocol, port

  Destination Prefix

  DSCP

- Miscellaneous

LSP statistics

LSP ”traceroute”

DiffServ over MPLS linux routers can then be used for setting up explicit paths. Per-flow reservation is presently not supported by the kernel and therefore, the reservation of resources is done on per-class basis (DiffServ classes i.e. EF, AF, BE) at the kernel for QMPLS.

Explanation on how the LSPs are set on the DiffServ over MPLS linux routers can be found from [27].

5.3 DiffServ configurations

Having explained on how to create a DiffServ over MPLS router using patched Linux kernel and setting explicit routes using MPLS using RSVP-TE, we now explain how to create DiffServ classes. There are 5 network classes that are implemented at the network level as follows:

- EF(Expedited Forwarding): This is the guaranteed service class that provides highest level of service to the flows and corresponds to 101110 DSCP value. This class of service is used for CBR flows that require high level of QoS.

- AF13 (Assured Forwarding): This class of service is used for VBR class of service and corresponds to next level of service.

- AFX3 (Assured Forwarding): AF23 and AF33 are next 2 classes of services used for PMM and PMC for TCP flows.

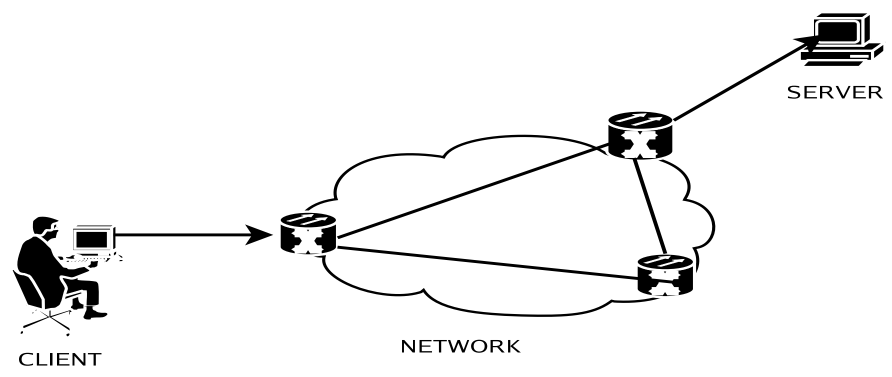

Figure 5.1: 3 node network DiffServ over MPLS TestBed

- BE: This is best-effort class traffic used for the rest of the traffic don'tthat require special QoS treatment.

5.4 Experimental Set-up

As already discussed, [27] explains how the redhat linux kernel can be patched to add DiffServ over MPLS functionality and RSVP daemon to the linux kernel. It also explains how we can set LSP (Label Switch Path) along some explicit route over the DiffServ over MPLS network. Previous section explains the set-up of DiffServ based service classes that are used for service differentiation and creation of network classes. Once we have understood these concepts, we can now create a network set-up that provides QoS guarantees to different application classes and thus, to different applications.

We implemented QMPLS on a test-bed using 3 DiffServ over MPLS Linux routers as shown in Fig 5.1 and separate client and server that request and sends flow traffic. The user specifies the QoS requirements on the application (a mail-application or a video streaming application) which are used by the admission control server to calculate the resources that might be needed to meet those requirements. The resources here include the path and the Diffserv class that would meet the requirements. In case, the requirements cannot be met, the request is refused admission. D-ITG [25] is used for end-to-end QoS measurement. To test the efficiency of the above proposed admission algorithm and architecture and *justify, that proposed traffic classes at the network level which have one to one correspondence with the application classes in our case improve the QoS for application flows for DiffServ over MPLS network, following tests are performed.*

5.4.1 Test-Case 1

These tests were intended to study the justification of having 2 proposed different QoS classes viz PCBR and PVBR for 2 different UDP flows (CBR and VBR) on a DiffServ over MPLS network. Following tests were performed:

Test 1

In test 1, 2 flows are transfered into the network. Flow 1 carries EF flow at the rate 100pkts/sec with each packet size equal to 256bytes. Flow 2 carries a poisson distributed traffic with the average rate of 16000 pkts/sec with uniformly distributed packet size between 40 B and 512B. Flow 1 simulates a CBR flow. Flow 2 represents VBR flows. Flow1 is sent for 120 secs. Flow 2 is sent after a delay of 30 sec and for a duration of 30 secs and the QoS for flow 1 (CBR) is observed.

- In the first case, the flows are transfered without any QoS policy. We observe following QoS for flow 1:

Total time = 59.946979 s

Total packets = 5995

Minimum delay = 0.000035 s

Maximum delay = 0.062872 s

Average delay = 0.000299 s

Average jitter = 0.000189 s

Delay standard deviation = 0.002888 s

Bytes received = 1534720

Average bitrate = 204.810321 Kbit/s

Average packet rate = 100.005039 pkt/s

Packets dropped = 5 (0.08 %)

- In the second case, the flows are transfered with DiffServ only implemented at the routers. Following QoS is observed for CBR flow:

  Total time = 59.998300 s

  Total packets = 5996

  Minimum delay = 0.000044 s

  Maximum delay = 0.099365 s

  Average delay = 0.003467 s

  Average jitter = 0.001460 s

  Delay standard deviation = 0.014096 s

```
Bytes received = 1534976
Average bitrate = 204.669266 Kbit/s
Average packet rate = 99.936165 pkt/s
Packets dropped = 4 (0.07 %)
```

- In the 3rd case, the flows are transfered with both DiffServ over MPLS working in effect. Following QoS is observed for CBR flow:

```
Total time = 59.950599 s
Total packets = 6000
Minimum delay = 0.000093 s
Maximum delay = 0.145902 s
Average delay = 0.003542 s
Average jitter = 0.001430 s
Delay standard deviation = 0.014492 s
Bytes received = 1536000
Average bitrate = 204.968761 Kbit/s
Average packet rate = 100.082403 pkt/s
Packets dropped = 0 (0.00 %)
```

From the above, we observe that the IP and DiffServ flow experience packetloss and bitrate when the additional VBR flow is sent to network for 30 sec. The DiffServ over MPLS (DS-MPLS) continue to retain its QoS performance parameters.

Our experiments have shown that it is difficult to obtain correct delay of the order of milli-second due to inaccuracy in the linux clock which used NTP to adjust times.

The graphs of the CBR flow obtained are shown.

Test 2

In this test, additional TCP flow of 1000pkts/sec with a packet size of 512 bytes was sent along with the above flows after a delay of 30 sec for 30 sec. We make following observations for CBR flow:

1. For IP network, the QoS observed for CBR flow is:

   Total time = 59.995827 s
   Total packets = 5762
   Minimum delay = 0.000046 s
   Maximum delay = 2.606972 s
   Average delay = 0.106489 s
   Average jitter = 0.002387 s
   Delay standard deviation = 0.413903 s
   Bytes received = 1475072
   Average bitrate = 196.689946 Kbit/s
   Average packet rate = 96.040013 pkt/s
   Packets dropped = 238 (3.97 %)

2. For DS network, QoS for CBR flow is:

Total time = 59.991223 s

Total packets = 5554

Minimum delay = 0.000045 s

Maximum delay = 5.556188 s

Average delay = 0.130001 s

Average jitter = 0.002857 s

Delay standard deviation = 0.761692 s

Bytes received = 1421824

Average bitrate = 189.604269 Kbit/s

Average packet rate = 92.580210 pkt/s

Packets dropped = 446 (7.43 %)

3. For DiffServ over MPLS network.

Total time = 59.935665 s

Total packets = 6000

Minimum delay = 0.000096 s

Maximum delay = 0.554820 s

Average delay = 0.005500 s

Average jitter = 0.000636 s

Delay standard deviation = 0.042086 s

Bytes received = 1536000

Average bitrate = 205.019833 Kbit/s

Average packet rate = 100.107340 pkt/s

Packets dropped = 0 (0.00 %)

From the above test, we again observe that the CBR flow for DiffServ over MPLS remains unaffected and results in no packetloss and better bitrate when additional flows are sent. CBR flow experience loss in case of both IP and DiffServ network. Delay in IP network is less dues to lesser processing at the routers and absence of queues.

Above results clearly show that the QoS received by each flow has improved when we separately do explicit routing for the 2 DiffServ classes (EF and AF). This means that we get improved QoS when we have 2 different application classes PCBR and PVBR for UDP flows mapped to different MPLS LSPs with the different DiffServ classes (EF and AF1x respectively). From the above tests following conclusions are drawn and verified:

5.4.2 Test-Case 2

Here, we studied the proposition of having 2 different application classes for TCP flows viz one for greedy TCP flows (knows as Premium Multimedia(PMM) by Aquila research) and other for short lived TCP flows (known as Premium Mission Control (PMC)) like DNS, database queries on Diffserv over MPLS network.

Following tests was performed: 2 TCP flows were sent. One at a rate of 12000pkts/sec and other at 1000pkts. We observe average bitrate of 47793.386223 (12,000pkts/sec) and 578.838888 Kbit/s (1000pkts/sec) on IP network, 49790.996537 Kbit/s and 624.025128 Kbit/s on DiffServ(DS) network and 50845.120436 Kbit/s and 640.054895 Kbit/s on DiffServ over MPLS network. Thus, we find that the flows receive better throughput in case of DiffServ over MPLS (DS-MPLS) than DS and IP network and it also shows that a DS-MPLS class.

5.4.3 Test-Case 3

In this subsection, the effect of separation of UDP and TCP classes is studied. For this following tests were performed:

In this test, 4 flows each corresponding to each application class is sent. First flow is UDP and is sent at the rate of 1000 pkts/sec with the uniform packet size of 256B. Second flow again is a poisson distributed UDP flow sent at the average rate of 1000pkts/sec with the size changing uniformly between 40B and 512B. Third flow is a TCP flow sent at the rate of 5000pkts/sec with the packet size being equal to 512B. Fourth is a TCP flow sent at the rate of 100pkts/sec and has a packet size equal to 80B. Each flow above corresponds to each application class. Flow 1 corresponds to CBR, flow 2 to VBR, flow 3 to PMM and flow 4 to PMC.

We observe that the packetloss in case of DiffServ over MPLS for flow 1 is 15.41%, for DiffServ packet loss is 15.55% and for IP it is 15.77%. For flow 2, loss over DS-MPLS is 15.64%, over DS 15.67 % and over IP is 15.99%. For flow 3, bitrate over DS-MPLS is 20478.944310Kbit/sec, over DS is 20480.075606 Kbit/s and over IP is 20480.060075 Kbit/s. Bitrate over DS-MPLS for flow 4 is 64.004470 Kbit/s, over DS is 64.000710 Kbit/s and over IP is 64.002353 Kbit/s.

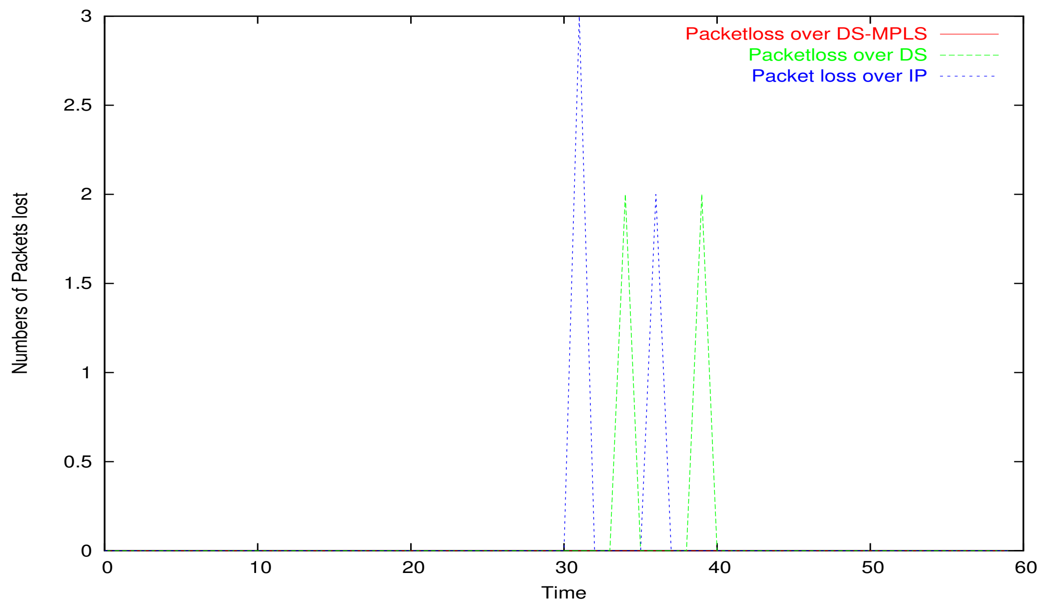

Figure 5.2: Packetloss of flow 1 in test-case 1

When we increase the rate of VBR flow from 1000pkts/sec to 4000pkts/sec, we observe that the bitrate of flow 3 for DS becomes 19949.464630 Kbit/s, for IP becomes 16757.225827 Kbit/s, for DS-MPLS it is 20480.062464 Kbit/s. Bitrate for flow 4 for IP is 63.999306 Kbit/s, for DS is 64.003400 Kbit/s, for DS-MPLS is 64.004384 Kbit/s. Packet loss for flow 1 over DS-MPLS is 24.85 %, over DS is 27.00 % and over IP is 26.52 %. Thus, we clearly observe that DS-MPLS shows lesser degradation in QoS than DS and IP for all the classes when the traffic rate of VBR (class 2) flow is increased.

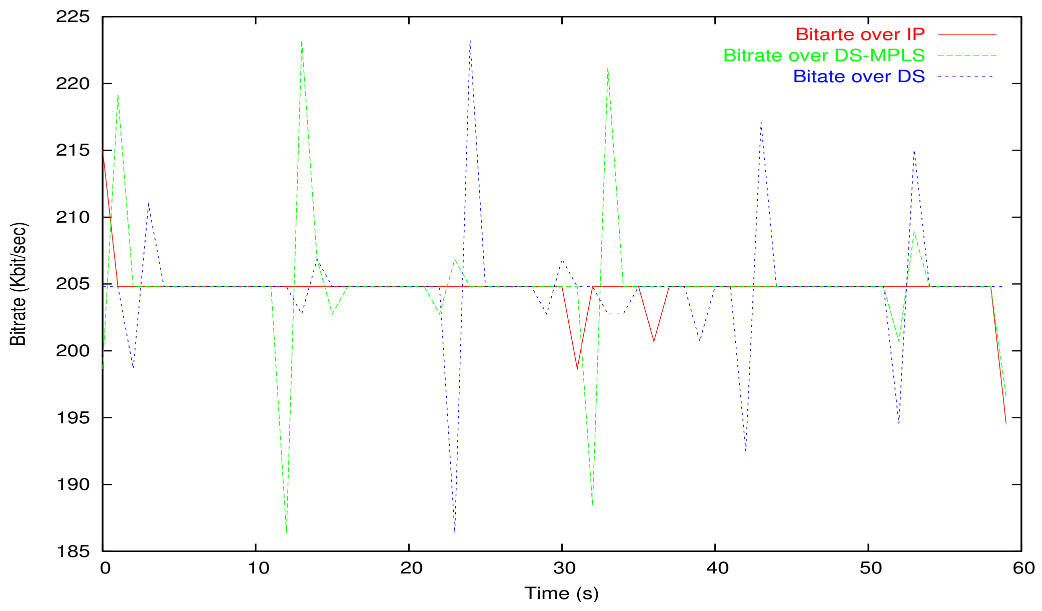

Figure 5.3: Bitrate for flow 1 in test-case1

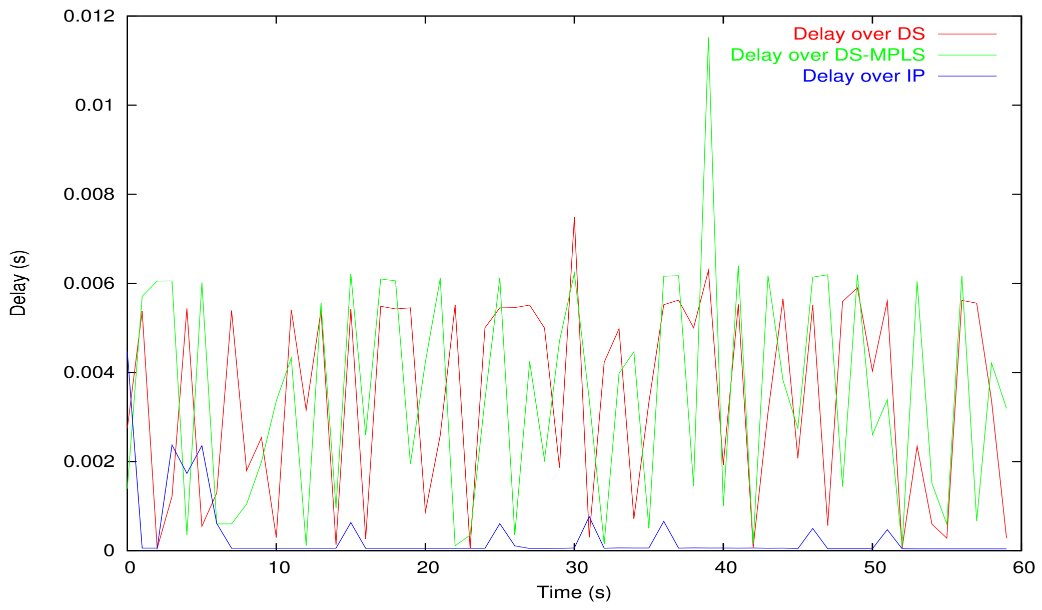

Figure 5.4: Delay for flow 1 in test-case1

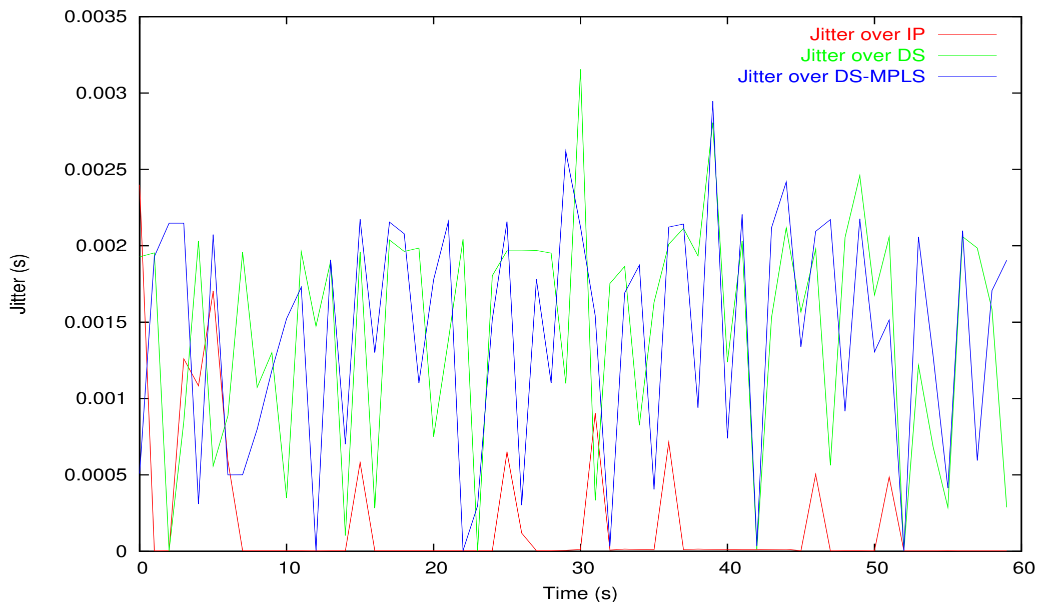

Figure 5.5: Jitter for flow 1 in test-case1

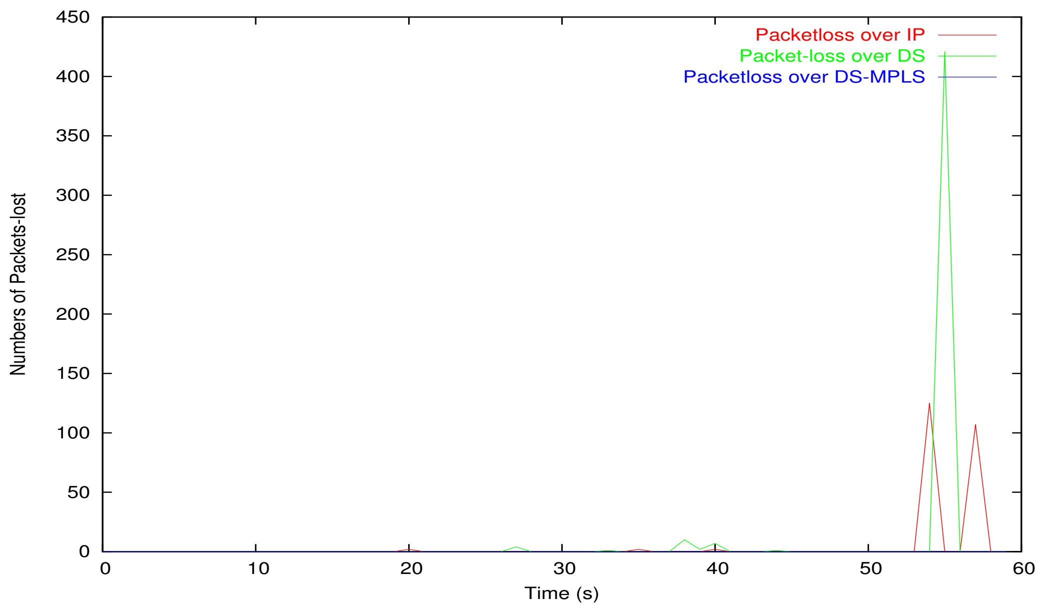

Figure 5.6: Packetloss of flow 1 in test-case2

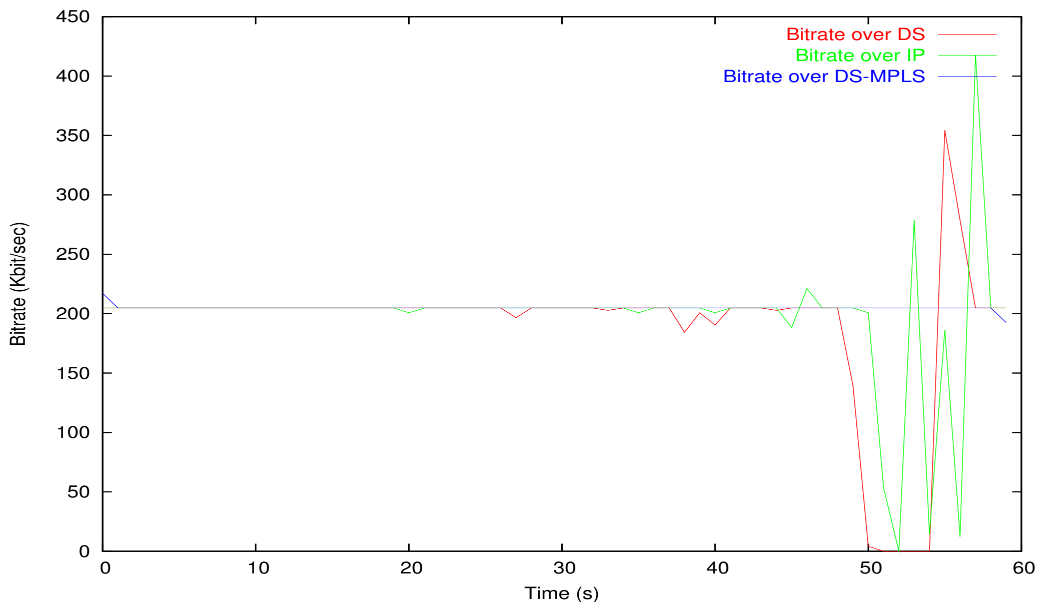

Figure 5.7: Bitrate of flow 1 in test-case2

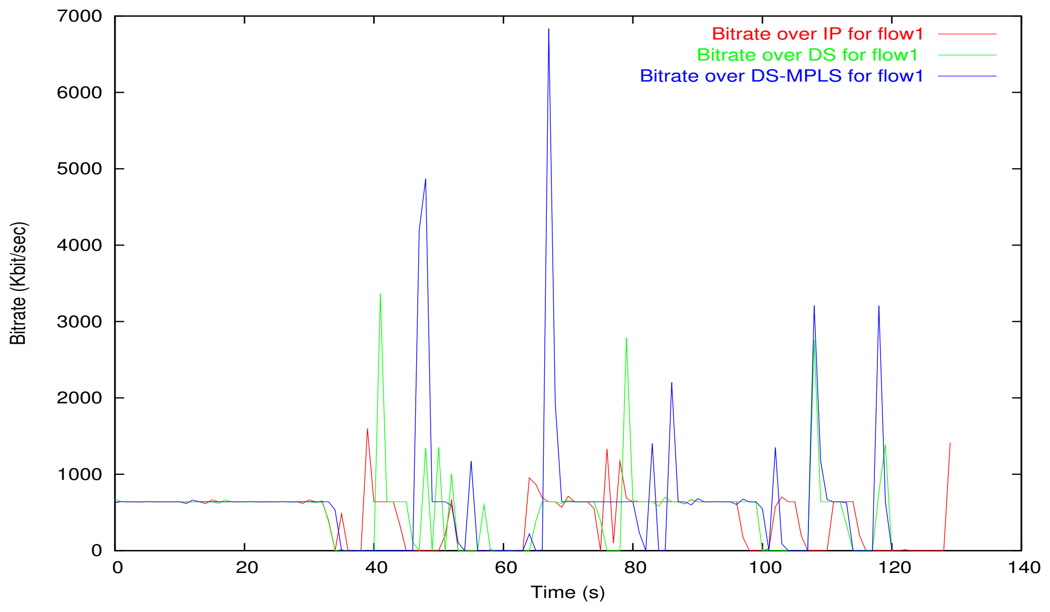

Figure 5.8: Bitrate of flow 1 in test-case3

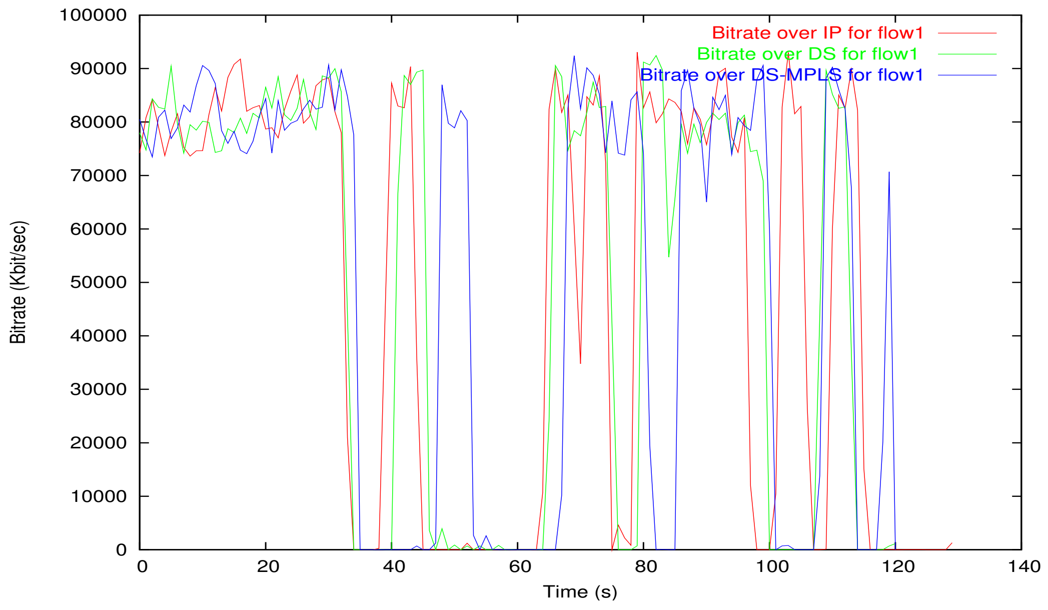

Figure 5.9: Bitrate of flow 2 in test-case3

Chapter 6

Conclusion and Future Work

At the core of the above problem lies the effect of explicit routing on different QoS parameter. We have proposed an architecture that does this. We have proposed 2 admission control algorithms one without preemption and one with preemption. During our work and research, it has become evident that no mapping mechanism to translate application QoS needs of the network to the underlying network configurations for the MPLS networks.We have dealt with the research problem of meeting SLAs on per application basis wherein the customer choses a particular SLA out of few offered by the service provider. Service Level Agreement (SLA) is a formal negotiated agreement between a service provider and a customer. When a customer orders a service from a service provider, an SLA is negotiated and then a contract is made. In the SLA contract, QoS parameters that specify the quality level of service that the service provider will guarantee are included. The service provider must perform SLA monitoring to verify whether the offered service is meeting the QoS parameters specified in the SLA.In order to do so, the service provider is able to systematically map such network performance metrics and data to application level QoS parameters.

As a future work, different MPLS TE (traffic engineering) techniques for effective QoS for different proposed QoS classes can be studied. As an example, currently in our architecture, explicit path for CBR flow is found by not considering other classes in the network. Effect of other classes on finding the path such that the QoS performance of CBR flow is not disturbed can be an interesting problem.

References

[1] Paul Ferguson, Geoff Hustion. Quality of Service. *John Wiley and Sons, Inc.* 1998. ISBN:0471243582

[2] Dinesh Verma. Supporting Service Level Agreements on IP Networks. *Macmillan Technical.* 1998

[3] Hyo-Jin Lee, Myung-Sup Kim and James W. Hong. Mapping between QoS Parameters and Network Performance Metrics for SLA monitoring. *Distributed Processing and Network Management. Dept. of Computer Science and Engineering POSTECH, Pohang, Korea*

[4] D.Davide Lamanna, James Skene and Wolfgang Emmerich. SLAng: A Language for Defining Service Level Agreement.Department of Computer Science University College London Gower Street, London.

[5] Currecoechea, A. T. Campbell, and L. Hauw. A Survey of QoS Architectures.*ACM/Springer Verlag Multimedia Systems Journal, Special Issue on QoS Architecture, 6(3):138.151.* May 1998.

[6] Frank Siqueiral. Quartz: A QoS Architecture for Open Systems. *Ph. D. Thesis, Department of Computer Science, Trinity College, Uni-*

versity of Dublin.http://www.inf.ufsc.br/ frank/papers/PhD-Thesis.pdf. December, 1999.

[7] Silvia Giordano, LCA – EPFL Stefano Salsano, DIE – University of Rome, Tor Vergata Steven Van den Berghe, IMEC Giorgio Ventre, University of Naples Federico II Dimitrios Giannakopoulos, National Technical University of Athens. Advanced QoS Provisioning in IP Networks : The European Premium IP Projects. *IEEE Communication Magazine.* January 2003

[8] European IST Project.Adaptive Resource Control for QoS Using an IP-based Layered Architecture (AQUILA). *World Wide Web. http://www-st.inf.tu-dresden.de/aquila/*

[9] Andrew T. Campbell.A Quality of Service Architecture.*Ph. d. Thesis. Computing Department, Lancaster University.1996*

[10] Jingwen Jin, Klara Nahrstedt. Specification Languages for Distributed Multimedia Applications: A Survey and Taxonomy. *IEEE MULTIMEDIA MAGAZINE. http://cairo.cs.uiuc.edu/publications/paper-files/ieeeimultimediajin.pdf*

[11] Zheng Wang. Architecture and Mechanisms for Quality of Service.*Morgan Kauffman*

[12] RBraden, D. Clark, S. Shenker. Integrated Services in the Internet Architecture: an Overview, RFC1633. *World Wide Web. http://www.ietf.org/*

[13] R. Braden, L.Zhang et. al.Resource Reservation P4 02/21/Protocol (RSVP), RFC 2205.*World Wide Web.http://www.ietf.org/*

[14] R J. Wroclawski. The Use of RSVP with IETF Integrated Services, RFC 2210. *World Wide Web.http://www.ietf.org/*

[15] R J. Wroclawski. Specification of the Controlled-Load Network Element Service, RFC2211. *World Wide Web. http://www.ietf.org/*

[16] S. Shenker, C. Patridge, R. Guerin.Specification of Guaranteed Quality of Service, RFC2212. *World Wide Web. http://www.ietf.org/*

[17] E. Rosen, et al. Multiprotocol Label Switching Architecture. *IETF RFC3031* January 2001.

[18] D.Awduche, et al. Requirements for Traffic Engineering Over MPLS. *IETF RFC2702.* September 1999.

[19] Le Faucheur, F., et. al., Requirements for support of Diff-Serv-aware MPLS Traffic Engineering, work in progress. http://www.ietf.org/

[20] ITU-T Recommendation Y.1541. Network Performance Objectives for IP-Based Services. May, 2002.

[21] N. Blefari-Melazzi, M. Femminella. Stateful vs. Stateless Admission Control: which can be the gap in utilization efficiency? *GLOBECOM 2002 - IEEE Global Telecommunications Conference*, vol. 21, no. 1, November 2002, pp. 2560 2564.

[22] Jianming Qiu Huai-Rong Shao Wenwu Zhu Ya-Qin Zhang. An End-to-End Probing-Based Admission Control Scheme for Multimedia Applications. http://csdl.computer.org/comp/proceedings/icme/2001/1198/00/11980170abs.htm.

[23] Aquila application specification language. http://www-st.inf.tu-dresden.de/aquila/files/application-profiles.htm

[24] Network Simulator. http://www.isi.edu/nsnam/ns/.

[25] http://www.grid.unina.it/software/ITG/

[26] Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications Koushik Kar, Murali Kodialam, Member, IEEE, and T. V. Lakshman, Senior Member, IEEE. http://www.ecse.rpi.edu/homepages/koushik/mypapers/jsac00.pdf.

[27] ds-mpls linux daemon and patch download from tequilla RSVP site. http://ds_mpls.atlantis.rug.ac.be

[28] "Specification of traffic handling for the first trial" Aquila Deliverables.