Université d'Ottawa · University of Ottawa

SUPPORT OF DELAY SENSITIVE APPLICATIONS OVER MPLS AND DIFFERENTIATED SERVICES ENABLED NETWORKS

BY

TAREK W. SAAD, B.ENG

A thesis submitted to the Faculty of Graduate And Postdoctoral Studies in partial fulfillment of the requirements for the degree of Master of Applied Sciences in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Information Technology and Engineering

Faculty of Engineering

University of Ottawa

ABSTRACT

The exponential growth of the amount of information exchanged over the Internet, and the emergence of new access technologies for high-speed Internet services—like DSL, Internet over cable, and Internet over satellite— have paved the way for new types of applications that demand performance guarantees beyond the conventional best-effort service. Telecommunication companies and Internet Service Providers (ISPs) alike are faced with the challenge of fully optimizing their networks for IP while supporting real-time applications that demand hard throughput and delay constraints on the network.

In order to be financially successful in this environment, service providers will have to support a variety of services and applications on a common packet infrastructure, carrying increased varieties of traffic with different performance characteristics and predictable levels of managed Quality of Service (QoS). Multi-Protocol Label Switching (MPLS) traffic engineering facilitates this task and allows the network to provide such QoS.

To this extent, the thesis proposes a framework for applying effective QoS techniques in a Diffserv MPLS network to ensure predictable levels of QoS at the network layer to delay-sensitive applications. This approach will enable network edge routers to discriminate against classes of traffic, and dynamically map it into pre-established Label Switched Paths (LSPs) that emulate classes of service (e.g. gold, silver, etc.). We investigate the several contributors of delays in a packet switched IP network in order to ensure minimal service end-to-end delay. We present a number of scenarios to prove our conclusions.

Our work is first simulated on a network simulation tool, Optimized Network Engineering Tool (OPNET) [16]. The second phase of the experimental work is to realize, and evaluate our model on an MPLS test-bed of networked PCs running on Linux Operating System (OS) and

connecting a number of Multimedia Streaming server/client applications. The Linux kernel is patched so it becomes Diffserv MPLS-enabled.

DEDICATION

To the loving memory of my beloved father (1932- 2001)

ACKNOWLEDGEMENTS

All praise is due to Allah, the lord of the Worlds, the most beneficent, and the most merciful to whom I owe all I have. Throughout the course of my research, I have gathered immense technical research skills and abilities with the backup, support, and patience of several individuals—to whom the least I wish is to express, is a sincere word of gratitude and recognition.

I would start by thanking my supervisors Prof D. Makrakis, and V. Groza for their support, technical assistance and advice, and most importantly for putting faith in me to conclude this work. I also would like to thank Prof. E. Petriu for partially funding my research.

Many thanks to all friends at the Broadband Wireless and InternetWorking Group (BWIWG) at the University of Ottawa for providing all the support and advice at times when I most needed it.

I also want to express the least of gratitude to my family whose faith in me has provided—and still does—the strength to carry on. Special appreciation goes out to my sister Mona, and her family for their encouragement and assistance in times when I most needed it.

Finally, special thanks to my wife, Hala, for her love, encouragement, and patience.

Thanks to everybody

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

AF	Assured Forwarding
BE	Best Effort
CBR	Constraint-Based Routing
CBQ	Class Based Queuing
CoS	Class of Service
CPF	Cheapest label-switched Path First
Diffserv	Differentiated Services
DSCP	Differentiated Services Code Point
EF	Expedited Forwarding
E-LSP	EXP inferred (PSC) LSP
ER	Explicit Route
FEC	Forwarding Equivalence Class
FTN	Forwarding to NHLFE
ILM	Incoming Lookup Map
Intserv	Integrated Services
IP	Internet Protocol
IPDV	Instantaneous Packet Delay Variation
IPMP	IP Measurement Protocol
IPPM	IP Performance Metric
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
LDP	Label Distribution Protocol
LSP	Label Switched Path
L-LSP	Label Inferred (PSC) LSP
LSR	Label Switching Router
MPEG	Motion Picture Expert Group
MPLS	Multiprotocol Label Switching
NGI	Next Generation Internet
NHLFE	Next Hop Label Forwarding Entry
NTP	Network Time Protocol
OPNET	OPtimized Network Engineering Tool
OSI	Open System Interconnection
OSPF	Open Shortest Path First

OTT	One Way Transit Time
OWD	One Way Delay
PHB	Per Hop Behavior
POP	Point of Presence
PSC	Per Hop Scheduling
PQ	Priority Queuing
QoS	Quality of Service
QoSR	Quality of Service Routing
RED	Random Early Detect
RIP	Routing Information Protocol
RSVP	Resource Reservation Protocol
RTCP	Real Time Control Protocol
RTSP	Real Time Streaming Protocol
SFQ	Stochastic Fair Queuing
SLA	Service Level Agreement
SLS	Service Level Specifications
SNTP	Simple Network Time Protocol
SP	Service Provider
TCP	Transmission Control Protocol
TCS	Traffic Conditioning System
TE	Traffic Engineering
TT	Traffic Trunk
TTL	Time To Live
UDP	User Datagram Protocol
VBR	Variable Bit Rate
VPN	Virtual Private Network
WF2Q	Worst-case Fair Weighted Fair Queuing
WFQ	Weighted Fair Queuing

Chapter 1

INTRODUCTION

1.1 BACKGROUND

Services offered on today's networks are rapidly evolving in scope and availability as new access technologies come into view- enabling multimedia interactive distance learning, virtual reality, video teleconferencing applications to be within the reach of almost all users. However, the momentum towards the convergence of these network services over a robust multi-service network is driving the Internet to cope with new realities. Historically, the Internet infrastructure and its protocols were designed and optimized solely for data transfer. Traditional routing protocols incorporating Interior Gateway Protocols (IGPs) such as Routing Information Protocol (RIP), Open Shortest Path First (OSPF), and Intermediate System to Intermediate System (IS-IS) [22][23]– and Exterior Gateway Protocols (EGPs), such as Border Gateway Protocol (BGP) [24], no longer represent an optimal, and in several cases not even acceptable, solution for a multi-service network. New solutions that classify critical data with specific QoS and Class of Service (CoS), and introduce more intelligence into the network—making it more adaptive to changes in traffic state—are surfacing up.

The unpredictable loss, delay and delay jitter along certain Internet paths can adversely impact the performance of real-time applications, such as audio and video conferencing. Such applications may need proper resource provisioning in the network to achieve acceptable end-to-end quality. There has been a significant research effort in changing the Internet architecture to one that can provide different service levels for specific QoS requirements. However, it remains an open question how to regulate the provisioning of resources or services to a particular group of users or hosts depending on the network conditions.

1.2 MOTIVATION

It is an inherently challenging task to provide strict deterministic network service guarantees to QoS sensitive applications using a non-deterministic IP network. The current routing technologies deployed in conventional networks utilize the best available path information based on the destination address; hence, the application data's attributes are not considered. Moreover, as the network grows, routers will need to handle even higher volumes of information, besides making forwarding decisions at each hop, inhibiting scalability and performance.

Different types of traffic have different delay, loss, jitter requirements. The Diffserv architecture differentiates between types of traffic that share the same link, and offers preferential treatment of some classes over others. MPLS provides for the possibility to differentiate between the paths certain types of traffic follow from a source to a destination. Combining the two technologies, we can provide a mechanism to dynamically define the path of certain "mission-critical" traffic with specific QoS requirements. For example, voice and video have become of the most important types of traffic carried on today's networks. For an IP network to be truly considered "multi-service", it must accommodate these types of traffic and their unique transmission requirements. Unlike data traffic, real-time traffic needs to travel persistently through the network without being subject to delay or packet re-ordering. A standard IP network, operating on a best-effort basis, is unable to guarantee such preferred treatment.

The research on providing service guarantees in packet switched IP networks focuses on a number technologies that were proposed by the Internet Engineering Task Force (IETF) [36] and can be classified into two fundamental streams:

- Services that allocates dedicated resources for exclusive use by individual micro-flows
- Services that classifies traffic into aggregate "classes" that share dedicated resources

The Resource reSerVation Protocol (RSVP) [37] and Integrated Services (Intserv) [40] provide the former, and Differentiated Services (Diffserv) [41] and MPLS [39] provides that latter. Each of these has a separate working group in the IETF.

While the Diffserv architecture is able to provide preferential treatment for certain packets over others inside the core of the network, by itself, however, it fails to provide strict deterministic guarantees on end-to-end performance. To provide tighter bounds of QoS that some mission-critical or delay-sensitive applications require, the Diffserv architecture can be augmented by a Traffic Engineering (TE) scheme that is resilient to changes in the network conditions.

MPLS is gaining significance as a transport networking technology in the future Next Generation Internet (NGI) thanks to its easy implementation of efficient traffic engineering [17][23]. TE aims at the ability to efficiently map available traffic onto existing network topology in a way that optimizes the utilization of network resources, and ensures QoS constraints are met. MPLS supports TE by allowing the node at the network ingress to specify the path that a LSP will take using explicit routing (ER) features. An explicit route is specified by the ingress as a sequence of hops that must be used to reach the egress, as opposed to traditional hop-by-hop routing schemes that are usually associated with packet-switch capable networks. When coupled with QoS, ER is usually referred to as Constraint Based Routing (CBR) [11]. ER can be used to optimize the utilization of network resources and enhance traffic-oriented performance characteristics.

MPLS and Diffserv [10][29] are emerging to be the choice for the network core infrastructure of the new multi-service network, making it possible to address many of the shortcomings associated with current IP routing schemes. An MPLS enabled network is able to provide low-latency and guaranteed traffic paths for real-time traffic, allocating delay-sensitive traffic Forwarding Equivalence Classes (FECs) corresponding to certain Classes of Service (CoS) that provide the service appropriate for this type of traffic.

1.3 OBJECTIVES

It is the objective of this work to identify the several contributing factors to delays in networks, and study the different technologies proposed so far to provide QoS guarantees to delay-sensitive applications in IP packet-switched networks. The advantages and drawbacks of each of the technologies are studied, and a scheme to integrate MPLS protocol in a Diffserv network to transport delay-sensitive traffic along several established ER-LSPs is suggested.

Further, we implement an adaptive mechanism to reroute mission-critical premium traffic from lower "best-effort" LSP onto higher priority "DS-enabled" LSP while tracking delay constraint and cost objectives. We adopt two approaches to carry out performance measurements along established LSPs, namely the active, and the passive approach.

A number of pilot simulations are initially run using OPNET tool to verify the validity of the proposal. Finally, the model is realized on a test-bed of networked PCs running Red Hat Linux version 7.2, kernel 2.4.17. The routers are configured to enable Diffserv and MPLS functions. The network is used to carry a number of various profiles of traffic (e.g. voice and video with several compression formats) from multiple sources to a multiple hosts.

We integrated into our testbed two types of video/audio multimedia streaming servers (Windows Multimedia Server [92], and VideoLAN [91] multimedia server). Several experiments are run transmitting a mix of delay-sensitive traffic generated by the video-streaming servers, and normal cross-traffic generated by traffic generators to study the effect on QoS parameters along different LSP configurations.

1.4 PREVIOUS WORK

Efforts on providing end-to-end QoS in IP networks have been extensively researched over the last few years. With the emergence of MPLS and Diffserv, researchers have moved towards implementing the advantages introduced along with these architectures. For example, there are several works that addressed the advantages of traffic engineering introduced with MPLS in IP networks such as those found in [10][11][22], and the possibility of integrating MPLS with Diffserv and/or Intserv architectures such as in [9].

In MATE [12] the author introduced an adaptive traffic engineering mechanism in an MPLS network that dynamically partitions incoming traffic among parallel LSPs to efficiently utilize the available network resources, and minimize congestion. However, this mechanism paid no attention to packet priority or class (Diffserv architecture) in the partitioning process, and concentrated on reaching a balanced load among the LSPs to achieve better packet delay, and loss performance.

Another project, the Traffic Engineering for Quality of Service in the Internet at Large Scale (TEQUILA) [49], was founded to study, implement and validate a set of service definition and traffic engineering tools to obtain quantitative end-to-end Quality of Service guarantees.

The use of active measurements for network performance monitoring have been often adopted by network service providers (NSPs) to monitor compliance with Service Level Agreements (SLAs) made with customers, and to trigger remedial actions when desirable. Several projects widely exist collect network statistics and monitor network health using active probes, such as: Active Measurement Program (AMP) [45], RIPE NCC's Test Traffic Measurements Service [59], Surveyor [55], the IETF draft on network performance measurements [43], etc.

MPLS for Linux is group of open-source projects to implement the MPLS stack, and portable versions of the signaling protocols (e.g. RSVP-TE, LDP, etc.) associated with MPLS, onto the Linux kernel. Several attempts currently exist to port MPLS to Linux (e.g. work supported by Ghent University, Information Technology Department (INTEC) [57] [58], and work supported by Fujitsu Laboratories Ltd. [95]).

1.5 THESIS ORGANIZATION

This thesis begins with an overview of the several technologies proposed by the Internet community as potential solutions to provide network end-to-end service guarantees. The various factors that constitute network end-to-end QoS are investigated.

Zeroing in on our goal, in Chapter 3, we present several illustrative examples and propose a solution to be implemented on a test-bed of Linux routers, and suitable for carrying delay-sensitive traffic in the presence of several other traffic background sources. We highlight the advantages of appending the Diffserv technology to MPLS networks to make efficient and cost effective traffic engineering decisions. We further present a number of QoS monitoring techniques (active probing, and passive policing) to track the performance along established LSPs of the MPLS network and help decide in re-routing delay-sensitive alternate LSPs when needed.

In Chapter 4, we present an overview of the networking functionalities present in Linux-based systems. We describe the steps followed in upgrading the Linux kernel to make it Diffserv and

MPLS-capable. We also present the topology of the MPLS network, the performance tools deployed, and the several models of sources of traffic used in carrying out the experiments.

In Chapter 5, we present a description of the several experiments that were conducted over the prototype network with a detailed analysis of the results. Finally, Chapter 6 concludes the thesis and sheds light on some future areas of research.

Chapter 2

END-TO-END QUALITY OF SERVICE IN IP NETWORKS

2.1 INTRODUCTION TO QUALITY OF SERVICE

The evolution of the IP-based Internet from a government-funded experiment to a worldwide commercial network, and the introduction of several novel Internet applications such as multimedia, IP-telephony, interactive distance learning, and distributed virtual environment applications has driven researchers towards finding solutions to guarantee end-to-end QoS. These applications are increasingly being used over the all-reaching best-effort Internet. At the present, there is peak demand for the development of QoS solutions that encompass the characteristics of the Internet as well as enterprise networks.

The sound notion of QoS essentially encapsulates several levels of architecture and range from QoS at the user, application, system, and finally the network level. In order to experience real and evident quality in the end-to-end service offered, these levels have to all cooperate to produce the overall outcome. In this thesis, however, the emphasis is on QoS at the network level to achieve better network end-to-end service. In what follows the term QoS will be extensively used to point to network level QoS.

One of the key technical issues with QoS is that it must be end-to-end supported to be effective. This is particularly true for two-way services such as IP telephony and video conferencing, which must have a bounded delay and delay variation to be of a usable quality. Hence, the availability of QoS mechanisms in all portions of the network is essential to ensure this type of

traffic gets the appropriate handling from source to destination. This is analogous to the claim that a chain is only as strong as its weakest link [8].

Although there is no formal definition for the term QoS, it is usually associated with a collection of service-performances that allow customer users to request certain predicable service levels. These include connection setup time, service availability (uptime), error performance, response time and throughput, percentage of packet loss due to network congestion, and the frequency of fault detection and correction, end-to-end delay, and packet delay jitter. These performance metrics tend to be extensively negotiated and agreed between a customer, and a carrier service provider in a service contract, often referred to as Service Level Agreement (SLA) [49], which effectively specifies the treatment a customer's traffic should receive.

The traditional best-effort Internet was not designed to support a specified, desired, or consistent levels of QoS to network traffic. In order to support the service constraints for these new applications, frameworks and protocols have been proposed to provide different levels of network service assurance at different costs. Mixing together economic needs, and the appearance of the new applications dictates that the best-effort model of the Internet has to be changed or, at least, improved.

The IETF has proposed, through its working groups, a number of different models and supporting technologies that attempted to offer protection for the real-time traffic from network congestion. These included Differential Services (Diffserv), RSVP, Resource Allocation Protocol (RAP), Multiple Protocol Label Switching (MPLS), and Quality of Service Routing (QoSR) [14].

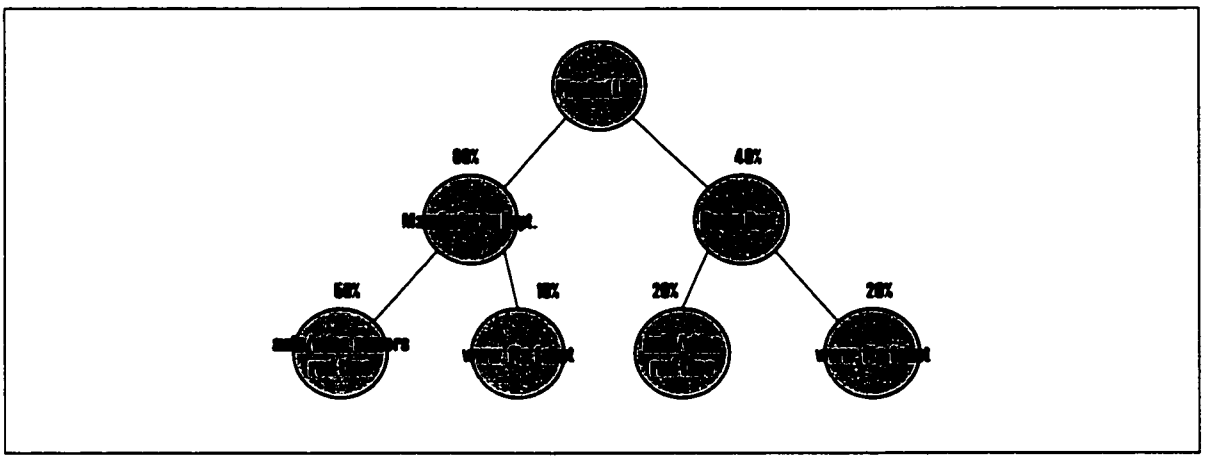

Figure 2.1 Example of a Physical Link's Bandwidth shared among aggregates of traffic

The purpose of QoS in networks has been to classify traffic into a subset of the overall set of traffic, then, assign the proper resources and apply certain policies and rules so as to provide predictive guaranteed services (as stated in the SLA documented parameters). For example, resources on an outgoing link can be divided among departments of a certain corporation, and further subdivided in each departments based on the application traffic type as shown in Figure 2.1. An example a QoS stack at the network layer is shown in Figure 2.2, whereby it is possible to have certain QoS constraints guaranteed by implementing one or more of these functions.

Resource Management
Resource Reservation
Traffic Engineering
Link Sharing
Congestion Control

Figure 2.2 QoS view at the network layer

2.2 QoS Building Blocks

Although QoS has many parameters associated with its definition, in our research we focus on 4 main factors that are mainly affected by statistical multiplexing of classes of traffic. These are:

1. Throughput
2. End-to-End delay
3. Delay jitter
4. Loss

Quality of service bounds can be characterized as deterministic, statistical or fractional. The general form of a deterministic bound is $v \leq b$, where the quantity v is a performance measure of a QoS parameter (e.g. latency, loss, etc.), and the bound b is a threshold of acceptable performance. Statistical bounds are of the form $\Pr(v \leq b) \geq B$, i.e. v is bounded by b with at least a probability B. This constitutes a "softer" bound with loose constraints on the underlying network, suitable for some applications that offer error correction mechanisms. Fractional bounds are more practical and easier to verify than statistical bound, for example, a bound of $C_A(v \leq b) \geq B$, $(B \leq A)$ states that the number of instances over an any A packets where v is bounded by b is at least B, a fraction of A..

2.2.1 Throughput

Throughput [101] is a generic term that usually refers to the end-to-end measurement of the net bandwidth between source and destination hosts. This could be easily identified in a TDM system, for example, where the throughput is effectively the bandwidth of the communication channel (e.g. a DS-3 circuit is 45 Mbps). In a multi-hop path, the throughput can be severely throttled by a single congested link. The *bottleneck bandwidth* is the bandwidth of the slowest link on a path, which could well be outside the control of one service provider. Moreover, due to congestion, some of the packets within a traffic-flow, traversing a number of hops along a path, may get dropped, resulting in some application re-transmitting the lost information. This leads to the definition of *Goodput* as the measure of the network's ability to transfer significant quantities of data with a single congestion-aware transport connection [101].

In statistically multiplexed networks (e.g. TCP/IP), there exists several ways to define and measure throughput on a certain path. These include:

- The packet or byte rate across the circuit
- The packet or byte rate of a specific application flow (i.e. per micro-flow)

- The packet or byte rate of host-to-host aggregated flows (i.e. per class of traffic)
- The packet or byte rate of network-to-network aggregated flows (i.e per VPN link)

Based on its capabilities, a router might be able to support the throughput of a class of traffic by the amount of bandwidth it reserves for that type. In the best-effort paradigm, routers do not specifically control the amount of bandwidth assigned for each kind of traffic. Packets are placed in a First-In-First-Out (FIFO) queue. Consequently, all packets, including high priority ones, are served in turn on a FIFO basis, and end up sharing the available link bandwidth. When faced with congestion, or buffer over flow, prioritized and ordinary packets get equal chance of being dropped. In case of TCP flows, applications will adapt to the loss rate reducing their transmission rate. UDP flows, however, maintain the user-specified rate, and consequently suffer from overly high loss rate when faced with congestion.

On the other hand, in routers that are capable of discriminating against types of traffic, it is possible to dedicate portions of the overall bandwidth of an output port to a certain class, give strict priority for a specific class over other classes, or give strict priority with a bandwidth limit (to prevent the starvation of the other classes).

2.2.2 Delay

Delay (or *latency*) is defined as the amount of time it takes a packet to cross from one point in a network to another point in the network. There exists several contributors to the overall One-way Transit Time (OTT) end-to-end delay a packet experiences as it traverses a network:

- Forwarding delay
- Queuing delay
- Propagation delay
- Serialization delay

2.2.2.1 Forwarding Delay

This is defined as the amount of time it takes a router to receive a packet, check its routing policies, makes a selection on: the next-hop, the output interface, layer-2 encapsulation type, and then begin transmitting the packet through an "un-congested" output port [28]. Usually, this

accounts for the least share of delay it takes a router to perform its basic function and is typically measured in tens or hundreds of μs. Forwarding delay is a function of the networking technology used. In routers that perform forwarding based on layer 3 of the OSI, this includes inspecting the IP header of incoming packets for the destination address, and looking up an entry in the routing table corresponding for the longest match.

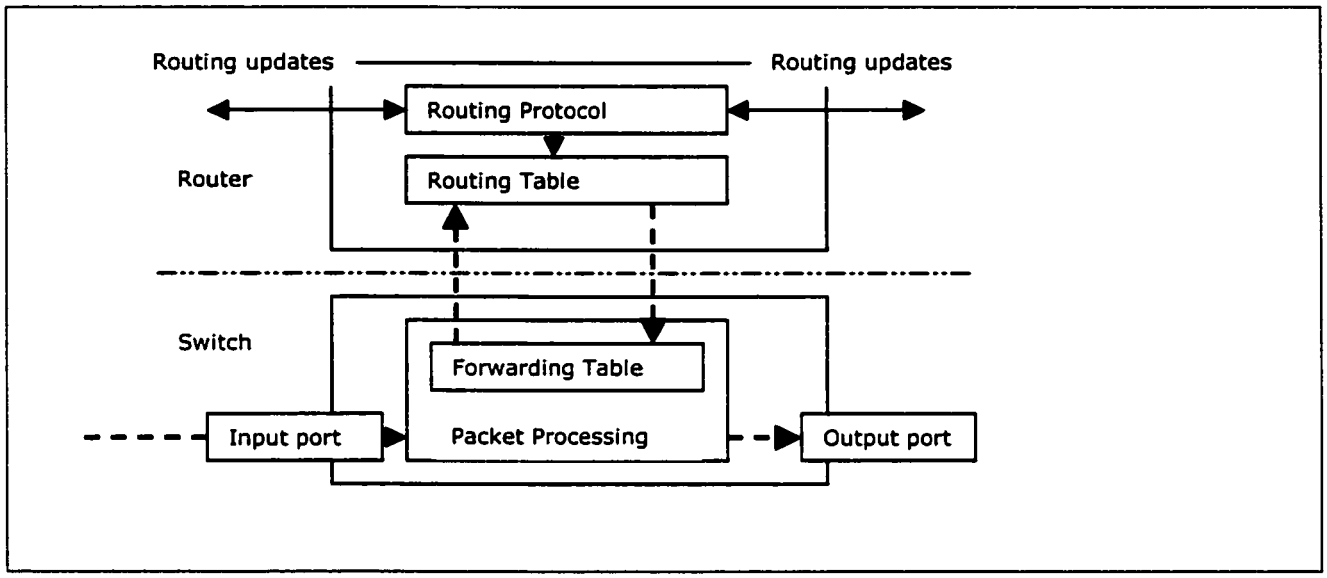

Figure 2.3 Control and forwarding routing functionality.

Switching however, includes functions performed only at layer 2 of the OSI. It involves inspecting the Datalink layer address, and swapping it with the destination next hop as in the case of ATM, Ethernet, or MPLS. Switching process is faster than layer 3 forwarding since it lacks layer 3 functionality, but suffer from scalability problems. Switching addresses are usually smaller than layer 3 addresses – for example, MPLS label is 20 bits as compared to at least 20 bytes of IP header in IPv4, and 40 bytes in IPv6; hence, involving fewer forwarding delays. It is also possible to minimize forwarding delay by implementing the switch fabric on fast hardware-based routers.

2.2.2.2 Queuing Delay

Queuing delay is the amount of time a packet waits in a queue after it arrives in a router, while the system performs statistical multiplexing and other packets are serviced. Thus, queuing delay ranges from zero – in the case of an un-congested link or empty queue— and the time it takes to transmit all other packets ahead of it. When a network is near maximum utilization, queuing

management functions and prioritization (e.g. Weighted Fair Queuing (WFQ), Worst-case Fair Weighted Fair Queuing (WF^2Q), Random Early Detect (RED), Class-Based Queuing (CBQ), Token Bucket First (TBF), Stochastic Fair Queuing (SFQ), etc) [48] are essential to ensure quality of service support. Hence, during periods of congestion, the queue memory management and queue scheduling disciplines are key to control the amount of queuing delay experienced by different classes of traffic placed in different queues. Queue management mechanisms operate within a single queue of a network node. When more than a queue exist several instances of the queue management can operate independently on each queue. For instance it is possible to have one queue with TBF management while another uses RED.

2.2.2.3 Propagation Delay

Propagation delay is the time it actually takes a frame to propagate through the communication link. It depends on the physical medium of the link (e.g. multimode fiber, twisted-pair copper wire, etc.), and the distance of the path involved. The propagation delay is based on the speed of signal propagation through the traversed medium. In certain cases, it can be significant part of the overall delay (for example, in satellite links). It is worth noting, however, that the propagation delay is independent of the channel transmission speed (i.e. a frame transmitted at 100 Mbps will experience the same propagation delay as one transmitted at 10Mbps) [69].

2.2.2.4 Serialization Delay

Serialization delay [100] is the time it takes to transmit an entire frame onto a communication link, after the router makes the decision to forward the packet, from first to last bit. Serialization delay is a function of the size of the packet and the link speed of the port. For example, a 100Mbps link experiences only one tenth of the delay of a 10Mbps link for the same transmitted packet. In actual IP networks, the only practical mechanism to reduce serialization delay is to reduce the Maximum Transfer Unit (MTU) in the network, or install higher-speed router interfaces.

Table 2-1 Serialization Delay—Packet Size vs. Port Speed [100].

Size\Interface	OC-1	OC-3	OC-9	OC-12	OC-48	OC-192
48 byte	0.2073 ms	0.0072 ms	0.0021 ms	0.0005 ms	0.0001 ms	< 0.0001 ms
256 byte	1.3264 ms	0.0458 ms	0.0132 ms	0.0033 ms	0.0008 ms	0.0002 ms
320 byte	1.6580 ms	0.0572 ms	0.0165 ms	0.0041 ms	0.0010 ms	0.0003 ms

512 byte	2.6528 ms	0.0916 ms	0.0264 ms	0.0066 ms	0.0016 ms	0.0004 ms
1500 byte	7.7720 ms	0.2682 ms	0.0774 ms	0.0193 ms	0.0048 ms	0.0012 ms
4470 byte	23.1606 ms	0.7994 ms	0.2307 ms	0.0575 ms	0.0144 ms	0.0036 ms
9180 byte	47.5648 ms	1.6416 ms	0.4738 ms	0.1181 ms	0.0295 ms	0.0074 ms

Another form of delay we mention here relates to the Link layer medium that the packet traverses, and is usually referred to as Media Access Delay. It is the time required for a head-of-the-line packet in the MAC transmit buffer to gain access to the medium. This is significant in contention-based networks, and is only caused by the medium competition and the fairness mechanism, not by the node's own traffic.

2.2.2.5 *Overall One-way Transit Time*

The One Way Transit Time (OTT) [51] of a packet can be generally modeled as the sum of several elements described by the following equation.

$$OTT = \sum_{j=1}^{n}(\frac{s}{b_j} + p_j) + \sum_{i=1}^{n}(q_i + f_i)$$

Here, s is the size of the packet, b_j is the bandwidth of link j, p_j is the cumulative sum of the propagation and serialization delay onto link j,, and q_i, and f_i are the queuing delay and the processing (forwarding) delay at node i [51].

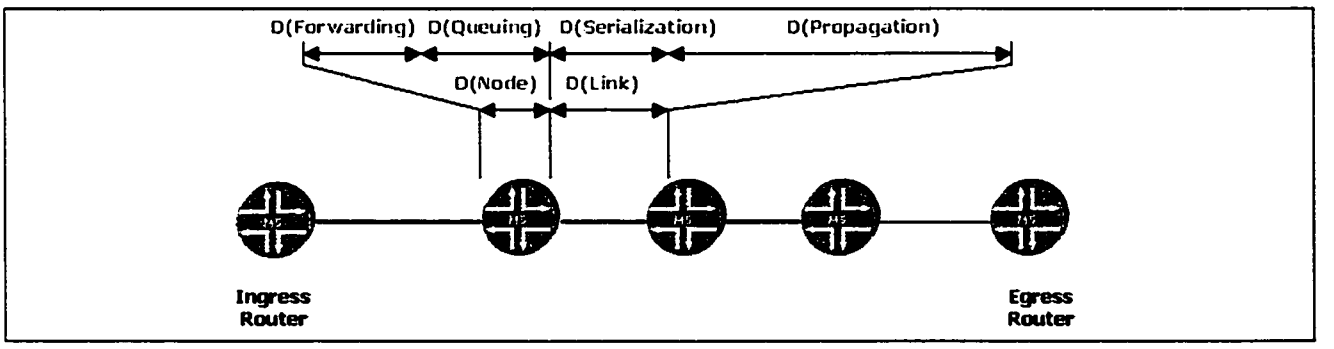

Figure 2.4 OTT overall end-to-end delay of a packet

Figure 2.4 illustrates the end-to-end delay experienced by a packet as the sum of the individual forwarding, queuing, serialization, and propagation delays occurring at each node and link in the network. It is worth mentioning that most of the research carried to provide bound guarantees on the end-to-end experienced delay primarily focuses on controlling the queuing, and forwarding delays, and sometimes directs towards evolving existing hardware and physical media to achieve faster output port interfaces, and lower propagation delays.

2.2.3 Packet Loss

Packets losses in IP networks occur for three main reasons:

1. A break in the physical link prevents the packet from reaching its destination.
2. A packet is diagnosed with a bad checksum error, and is discarded.
3. A packet is dropped from the router queue due to network congestion, and buffer overflow.

Thanks to the advanced modern technology, most of the packet losses due to physical link breakdown, and checksum corruption – with the exception of wireless technologies-- are recoverable. Hence, the primary cause of packet loss in IP networks is due to buffer overflow resulting from traffic burstiness, or network congestion. IP networks carry variable traffic loads, and at times the traffic gets very bursty exceeding the capacity that can be handled by some nodes in the network. This leads the nodes to drop excess packets. Recovery from this kind of loss in TCP/IP networks is left to the transport protocol incase of TCP, and to higher layers in case of UDP.

It's worth noting that packet loss sometimes plays an essential and positive role in TCP/IP networks. TCP protocol understands that "a packet drop" means congestion is present at some point in the network, and consequently reacts to the congestion by temporarily reducing the transmission rate of the flow. Eventually, all TCP flows will eventually settle on the maximum bandwidth they can get across the network without experiencing sustained congestion. This results in fairness to multiple TCP flows sharing the same path.

In IP Diffserv architecture, it is possible to group packets from several sources with certain priorities, and classify them with higher/lower drop precedence over other groups. This way priority classes suffer less losses than other best effort classes.

2.2.4 Jitter

Jitter [6] is the variation of delay experienced by consecutive packets that are of the same flow over time. There exist several definitions *Jitter*. Some authors understand jitter as the difference between the longest and the shortest delay in some period of time. Others define jitter as the maximum delay difference between two consecutive packets in some period of time [6].

Instantaneous Packet Delay Variation (IPDV), formally defined by the IPPM working group Draft [64], is based on one-way delay measurements and it is defined for (consecutive) pairs of packets. The measurement of a single IPDV requires at least two packets, and is defined as the difference of delays experienced by the consecutive packets. According to common usage, IPDV-jitter is computed as:

$$IPDV = D_t - D_{t-1}$$

$$IPDV\text{-}jitter = | IPDV |$$

Another definition of Jitter is standardized in the specification of Real Time Control Protocol RTCP [27] as a smoothed function of the delay differences between consecutive packets over time.

The main source of jitter in IP networks corresponds to the variation in the queuing delay experienced by consecutive packets in a specific flow. Another cause of jitter is that packets of the same flow may follow different physical paths, and consequently experience different delays to their destinations. This further results in having packets arriving out of order.

The effect of jitter on voice and video applications can result in jerky and uneven quality to sound or image. Jitter can be controlled by properly provisioning the incoming traffic to the network (e.g. queuing, scheduling, conditioning), or by a short playback buffer at the destination host that buffers the packets before playing them back as smooth stream.

However, there are other types of applications (such as those that run over TCP) for which jitter is not a problem. Also, for non-interactive applications, such as pre-encoded streamed voice or video, jitter does not present serious problems, because it can be overcome by using large playback buffers.

2.3 INTEGRATED SERVICES OVER IP

The Internet Integrated Services, developed within the IETF by working groups such as INTSERV [40] and RSVP [37], specifies extensions to the IP architecture which allow applications to request and receive a specific level of service from the network, as alternatives to the current IP best-effort service. The work of these groups has resulted in technology-independent protocols and specifications.

The proposed model included the Control Load (CL) service, Guaranteed (G) service, and Best Effort (BE) service to define QoS along the network [37]. The Guaranteed service is defined for applications with rigid end-to-end constraints, and ensures that packets arrive within the guaranteed delivery time and will not get discarded due to buffer overflows. The Controlled-Load service, on the other hand, is defined for applications with looser performance criterion, and offers a commitment to a service similar to that provided to best-effort traffic under lightly loaded conditions. The Intserv is then an extension of the original Internet architecture to support new types of real-time applications, and to improve the network manager's ability to control the network resources.

2.3.1 Resource ReSerVation Protocol (RSVP)

RSVP [37] is a signaling protocol that is used by hosts to request QoS from the network. It is used to deliver QoS requests to all nodes along the paths of flows and to create and maintain states relative to the different requests. RSVP is a receiver-based protocol, where a receiver initiates the request for services in response to a request from a sender. Hence, a sender starts an RSVP session by sending a *'Path'* message, whose function is to find a feasible path through the network for a particular flow and to bind this route for use by RSVP. In terms of QoS, the receiving host responds by sending reservation-request ('*Resv*') messages specifying the class of service it can support. These reservation messages are communicated all through the network elements in the reverse path until the sending host receives them. RSVP attempts to make a resource reservation at each network node the application flow will traverse on its path.

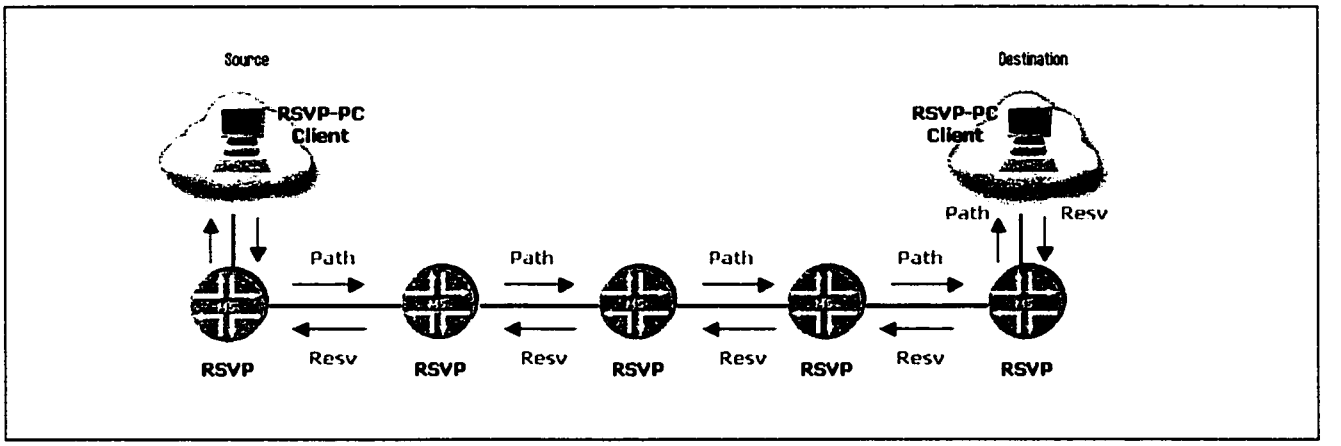

Figure 2.5 RSVP reservation through path and reserve message communication

The RSVP framework mandates that *a service sender* (*e.g.,* a video server) send out special *path finding* messages (*'Path'*, sender-receiver) periodically toward the service receiver (if unicast) or all potential receivers (if multicast). These *'Path'* messages and the data packet flow are delivered to the receiver(s) along the same routing path(s). *'Path'* messages not only implicitly discover the routing path(s) but also collect the QoS information along the path(s). An RSVP session is defined by the triple: (destination address, protocolId, destination port) [37].

The protocol defines objects such as Tspec(Path), which represents the traffic characteristics, and Adspec(Path), which represents the minimum resource level available for the routing path. Sender characterizes outgoing traffic in terms of upper and lower bounds of bandwidth, delay and jitter. The Path message contains this *traffic specification* (TSpec) information to the unicast or multicast address.

To make a resource reservation, receiver sends a *'Resv'* (reservation request) message "upstream" toward the sender along the reversed path. In addition to the Tspec, the reserve *'Resv'* message includes a *request specification* (Rspec) that indicates the type of Intserv required (Controlled Load or Guaranteed) and a *filter specification* (filter spec) that characterizes the packets for which the specification is being made (e.g. transport protocol and port number). Hence, the tuple (Rspec, and filter spec) define the flow descriptor entity which routers use to identify each reservation or flow [40].

Upon receiving *'Resv'* messages, an intermediate RSVP-enabled router on the path makes a decision whether to grant the reservation request according to its own policies. The reservation for a

new session will not be successful unless all of RSVP-enabled routers along the path grant the reservation requests.

RSVP takes a "soft state" approach to managing the reservation state in routers and hosts. RSVP's soft state is created and periodically refreshed by sending '*Resv*' messages along the reversed routing path toward the sender. Upon receiving a '*Resv*' message, each intermediate node refreshes the reservation state if it already exists, or creates it if it is a new reservation. Once a reservation is created, or after the refreshing of the reservation state the node forwards the request upstream. If an attempt to create a new reservation fails an error message is returned back to the receiver. The flowspec (flow specification) is used to set parameters in the node's packet scheduler or other link layer mechanism, while the filter spec is used to set parameters in the packet classifier.

The RSVP connection is deleted if no matching refresh '*Path*' or '*Resv*' message arrives before the "time-out" interval elapses. The connection may also be torn immediately by an explicit "tear down" message.

2.3.2 Issues With RSVP

Although the Intserv approach was able provide customer applications with the necessary end-to-end performance guarantees, yet several drawbacks to the overall deployment of this approach have emerged. First there were migration concerns. To provide QoS, the RSVP protocol required a high degree of cooperation between all the components along the path of the reserved stream, with the connection failing with any unsuccessful attempt to make a reservation at any of them. To be able to communicate traffic specifications to routers and initiate the reservation process, hosts had to be RSVP-enabled with specific APIs that network applications can invoke. This introduced more overheads at the host side especially on the newly introduced generation of small computers such as handheld PCs.

Second, issues related to security as to who can have access to resources in the network and as to how much resources a customer can exactly occupy have risen (e.g. user authentication, denial of service, etc).

Third, since it is impossible to deploy RSVP (or any new protocol) all at once throughout the entire Internet, RSVP had to cope with arbitrary clouds that were non-RSVP-enabled. To provide

connectivity, these routers had to forward *'Path'* and *'Resv'* messages onto remote RSVP-enabled sites. However, even though RSVP operates correctly through a non-RSVP cloud, the intermediate cloud that does not support RSVP is unable to perform resource reservation, and this will greatly alter its ability to provide QoS guarantees to the user [40].

Fourth, the Intserv architecture exhibits scalability problems in large networks, overloading control traffic by its soft state information. Moreover, RSVP's reliance on end-to-end per-flow state and per-flow processing on every node on the way makes it hard to deploy in large networks where flows number in tens of thousands. This pushed the research community to move on to a more coarse-grained approach.

Another downside in this scheme arises from the over-reservation of resources at components along the path to accommodate for worst-case situations. This can lead to low utilization if streams do not send the level of traffic they have reserved resources for. A sufficient level of best-effort traffic can remedy this problem by making effective use of available over-reserved resources.

With this in mind, the IETF has put effort to enhance the architecture of the RSVP protocol, and to reduce the effect of these drawbacks in implementing RSVP on a large-scale. Several proposals have been made to reduce RSVP's "chattiness," use it for tunneling, and augment it to support aggregated traffic flows.

Extensions to RSVP– RSVP Traffic Engineering (RSVP-TE) [18] – were introduced to enable support MPLS environment to carry explicit routing and label binding information, rather than QoS information. Here, when using RSVP to set up explicitly routed LSPs or tunnels, network operators get the option of allocating resources along that path using standard RSVP reservations and Intserv service classes.

In addition to tunneling support, RSVP has been enhanced so that a single reservation can aggregate other RSVP reservations. An aggregate reservation would be able to carry packets from a large number of flows that belong to the same traffic class or otherwise require similar treatment. Presently, there is a movement towards using Diffserv) markings to identify aggregated traffic; that is, traffic flows with the same Diffserv marking could be handled by an aggregated reservation.

2.4 DIFFERENTIATED SERVICES OVER IP

The shortcomings of the Intserv architecture led the IETF to define a more coarse-grained approach for managing traffic. The Differentiated Services (Diffserv) [30] architecture was defined as a potential replacement for the inscalable Intserv approach. Service differentiation is desired to accommodate heterogeneous application requirements and user expectations, and to permit differentiated pricing of Internet service. A Diffserv domain is set of nearby DS-enabled routers that operate with a common service provisioning policy and set of Per Hop Behaviors (PHB) groups implemented on each router.

2.4.1 At the Boundary of a DS domain

The Diffserv architecture places the complicated functionality on the edge of the network domain, while keeping the network core simple. Edge devices maintain all user traffic profiles and monitor incoming packets to ensure that it conforms to the respective profile.

2.4.1.1 Diffserv Classes

The Diffserv model introduces a mechanism to aggregate micro-flows into a number of predefined classes at the network edge. The architecture defines three classes of service: EF (Expedited Forwarding) [26], AF (Assured Forwarding) [63], and BE (Best Effort).

The EF service class was defined by the IETF in the context of Diffserv architecture to carry traffic with low delay and virtually no loss without per-flow queuing. As discussed earlier in chapter 2, loss, latency and jitter are mostly due to queuing experienced while traversing nodes in the network. Queues arise when traffic arrival rate exceeds departure rate at some node. Therefore providing low loss, latency and jitter for some traffic aggregate is only possible if the aggregate of traffic requiring EF treatment receives a service rate that is at least the total bandwidth requirements of all flows in the aggregate at this hop.

The AF class has four subclasses, AF1-AF4, each with its own forwarding priority and three levels of drop precedence. It provides a minimum bandwidth guarantee as well as efficient utilization of excess bandwidth. In the AF service edge devices allow more packets than specified in the profile to be injected as long as they are labeled with a mark specifying high drop preference. Core devices

are guarantee forwarding of in-profile packets; however, out-of profile packet forwarding depends on the availability of resources.

The Diffserv architecture defines a Diffserv fielding the IP header. This supersedes the Type of Service (ToS) octet in IP-V4, and the Traffic Class octet in IP-V6, and is used to make per-hop behavior (PHB) decisions about packet classification and traffic conditioning functions, such as metering, marking, shaping, and policing. The six most signification bits of DiffServ field are referred to as the Differentiated Services Code Point (DSCP), and the last two bits – originally left unused (Currently Unused (CU))– are now used as Early Congestion Notification (ECN) bits [25].

2.4.1.2 Diffserv Marking, Shaping, and Policing

Marking is a mechanism assumed by edge DS-nodes in a DS-domain to aggregate incoming flows, and group them into a set of pre-defined PHBs (e.g. EF, AF, and BE) based on the IP-defined DSCP present in the packet's IP header (see Figure 2.6). Core routers identify packets based on DSCP field in the IP header, and use it to select a PHB behavior for that packet. The default DSCP is (000000) and class-selector DSCPs are backward compatible with IP precedence.

Shaping is usually referred to as the technique of adapting outgoing traffic at a DS-edge router to certain traffic profiles and parameters negotiated in the SLA. Effectively, a shaper stores incoming traffic and releases it to the network after conforming it to certain parameters defined by Traffic Conditioning Specification (TCS) between an upstream DS domain/network, and its downstream neighboring DS domain.

Policing is typically implemented on traffic incoming at the DS edge network and it usually concerned only with applying policing functions on packets that exceed the agreed rates. From the technical point, shaping and policing functions are identical.

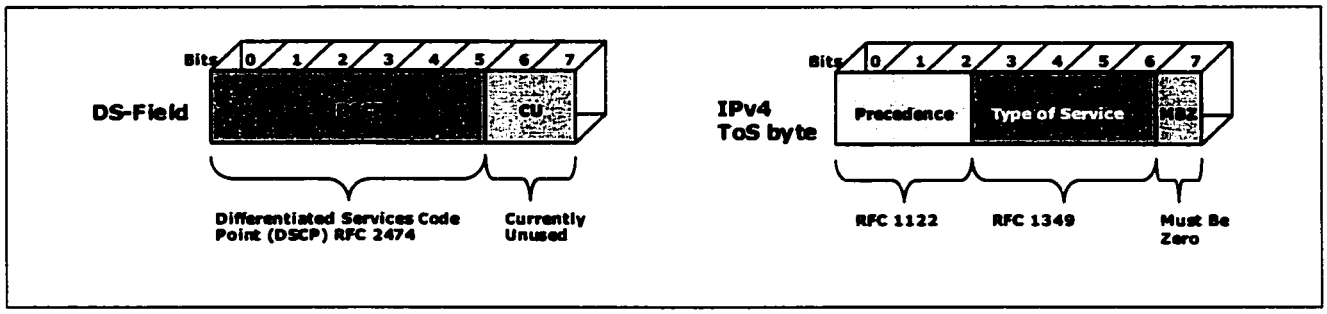

Figure 2.6 (Left) The DSCP field of the IPv4 header. (Right) The TOS field of IPv4 header.

Usually, Service Level Specifications (SLS) [52] specify the domain responsible for marking or "coloring" packets into predefined DS Behavior Aggregates (BAs). Packets that are colored by the end host or upstream domain pose less potential load on the downstream domain, requiring simply remarking and/or policing, to ensure the BA complies with the TCS. In cases where DS ingress routers connect to an upstream non-DS enabled domain, the ingress routers should be able to perform all necessary traffic conditioning and marking.

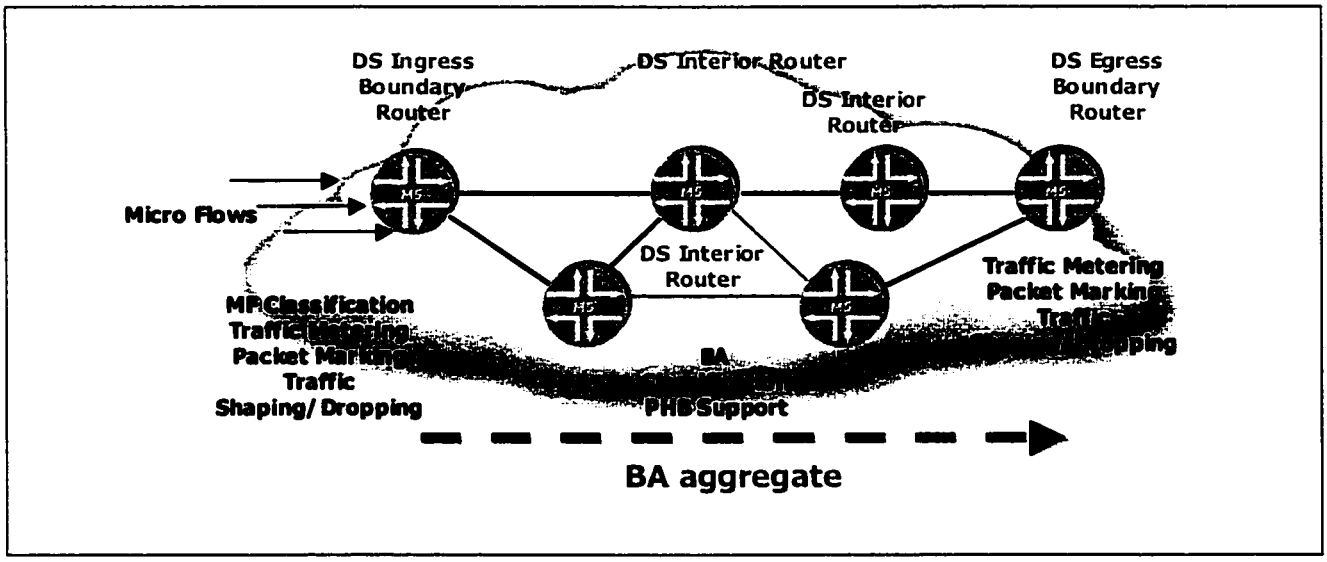

Figure 2.7 Building components of the Diffserv architecture and their functions

2.4.2 In Interior DS Nodes

The DSCP, found in the IP header, is polled at each Diffserv-enabled router on the way to invoke different PHBs. A PHB is a qualitative description of the forwarding behavior applied to a specific BA. PHB groups are designed having domain local importance or an end-to-end importance. As such, the PHB defines the way a node will allocate resources among BAs in a hop-

by-hop manner. Classifiers at core intermediate nodes classify traffic based on the DSCP forming purely BAs, or based on multiple fields in the IP header.

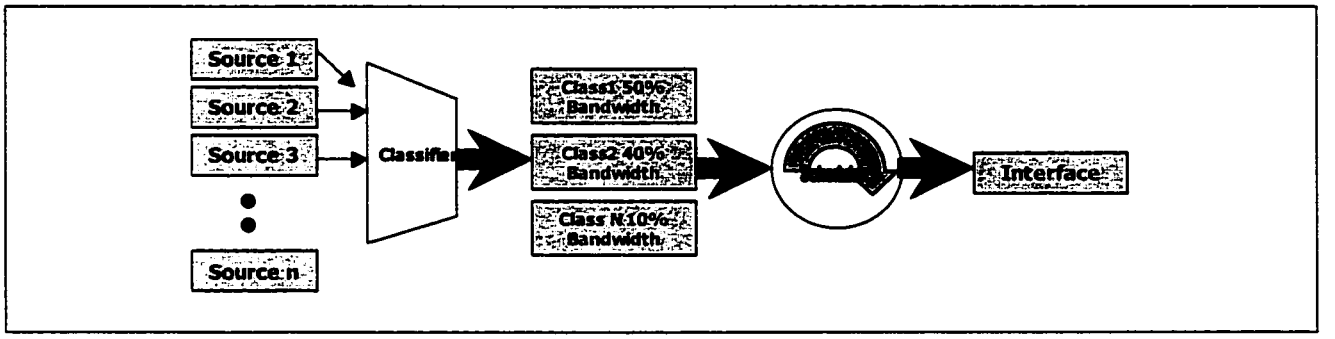

Figure 2.8 Sharing of the NIC's bandwidth between different classes

As an example, multiple flows from sources can arrive at an intermediate DS node. The classifier would classify the sources into classes (based on the DSCP), and then they are placed in the associated queues. These queues should be carefully defined to share the overall output port interface buffer. Packets that fall into a queue that is already full are either dropped or remarked to a lower priority class.

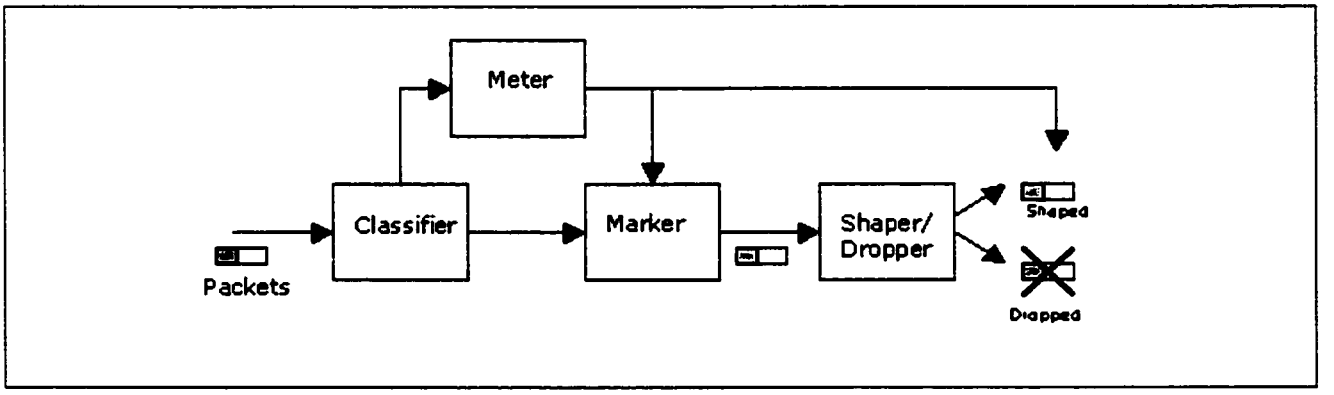

Figure 2.9 Components of a DS-enabled node

The DS architecture achieves scalability by implementing complex classification and conditioning functions – e.g. packet classification functions, and traffic conditioning functions only at network boundary nodes, and by applying per-hop behaviors to aggregates of traffic that were appropriately marked using the DS field in the IPv4 or IPv6 headers [30]. Core Diffserv-routers need only be concerned and forwarding treatments with PHB classification, possibly impose different traffic scheduling schemes (e.g. CBFQ, WFQ, WF^2Q, RED).

Hence, the amount information stored at each core DS-enabled router comes down to the number of defined classes as opposed to the number of micro-flows, as in the case of Intserv model. It is worth noting that this architecture works in asymmetric mode, allowing service differentiation only in one direction.

2.5 MULTI-PROTOCOL LABEL SWITCHING

Multiprotocol Label Switching (MPLS) [19][54] has emerged as a means to purposefully route or engineer traffic across a network. It is a layer 3 switching technology that aims at improving packet forwarding mechanism in the core of the Internet backbone by defining short fixed length labels and associating them with packets. This is in contrast with the network-layer address with variable length match. While not a QoS technology per se, MPLS includes QoS support as an outgrowth of its traffic engineering capabilities.

In our proposed solution, we leverage the growing migration of networks to Multiprotocol Label Switching (MPLS) to provide flexible routing and the ability to perform network traffic engineering [11][17]. MPLS has emerged as the preferred technology for providing QoS, traffic engineering and Virtual Private Network (VPN) capabilities on the Internet. MPLS provides a circuit-switching service in a hop-by-hop routed network. It achieves this by grouping related packets by assigning them a common, fixed-size label. Multiprotocol Label Switching is novel networking protocol that was developed to emulate connection-oriented networks – e.g. Frame Relay, ATM, etc.-- into IP networks to achieve levels of QoS for different classes of traffic. The name Multiprotocol comes from the fact that MPLS is capable of running over ANY network-layer protocol – e.g. IP, IPX, AppleTalk, etc.

2.5.1 Overview

Traditionally, a packet of a connectionless network layer protocol is examined at each subsequent router – from a source to a destination—to make an independent forwarding decision. That is, the router extracts each packet's addressing information (for example, the destination IP address in order to perform a longest match check), and runs a particular network layer routing algorithm to independently infer the next hop decision. This is usually referred to as the destination based forwarding model. Although this routing scheme proved to be adequate in the past, the need

to overcome its rigidity, and have more flexible control over how traffic is routed is growing to be a major concern. Moreover, it becomes more and more crucial to have a routing system that can support graceful evolution to accommodate new and emerging requirements.

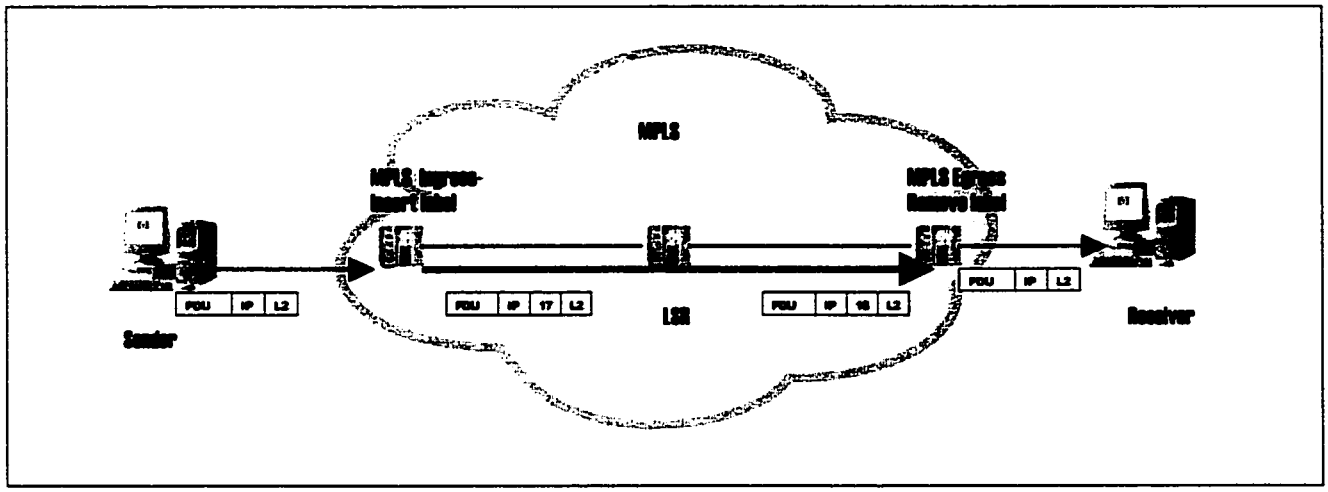

Figure 2.10 IP packet trip across the MPLS cloud.

MPLS integrates layer-2 switching speed, and layer-3 routing and addressing of the IP layer to create a scalable, IP-aware network that responds differently based on the type of traffic it carries. MPLS introduces a path oriented connection abstraction into the IP architecture. MPLS contains forwarding information for IP packets that is separate from the content of the IP header such that a single forwarding paradigm (label swapping) operates in conjunction with multiple routing paradigms. The basic operation of MPLS is to establish LSPs through the network into which certain types of traffic are directed.

An LSP is defined as a connection-oriented path from the ingress node to the egress node of an MPLS domain. A traffic trunk, on the other hand, is an aggregation of flows of the same class of service that effectively take the same LSP. It follows that all packets of the same trunk have the same labels, and the same Diffserv class-- i.e. EXP for EXP-Inferred LSP (E-LSP), and label for Label Inferred LSP (L-LSP) Per hop Scheduling Code (PSC) [29].

In an MPLS network we can define explicit paths that can be used to optimize the utilization of network resources and enhance traffic-oriented performance. For example, multiple paths can be defined and utilized simultaneously to improve performance from a given source to a destination. It is also possible to let IP routing layer-3 discover the network topology in which case layer 3 routing

IGP protocols (e.g. OSPF, IS-IS, etc) routing table database are used to define LSPs. Packets are assigned to Forwarding Equivalent Classes as they enter the MPLS domains, and are forwarded based on swapping labels as they traverse the network without any subsequent IP header analysis. A key feature in MPLS is to separate network control and data forwarding. This makes MPLS extensible to many environments and able to co-exist over a variety of link layer topologies – e.g. Ethernet, Frame Relay, ATM and PPP, and even over SDH and Optical.

MPLS is even gaining more significance as a transport networking technology in the future Next Generation Internet (NGI) thanks to its easy implementation of efficient traffic engineering [17][23] for the rapidly increasing Internet traffic. The simplicity of "label switching" forwarding paradigm enables improved forwarding performance and reduces the forwarding delay, while maintaining competitive price/performance.

By associating a wide range of forwarding granularities with a label, the same forwarding paradigm can be used to support a wide variety of routing functions, such as destination-based routing, multicast, hierarchy of routing knowledge, and flexible routing control. Finally, a combination of simple forwarding, a wide range of forwarding granularities, and the ability to evolve routing functionality while preserving the same forwarding paradigm enables a routing system that can gracefully evolve to accommodate new and emerging requirements.

2.5.2 The Development of MPLS

MPLS is a mere development by the Internet community. It has its origins attributed to a number of proprietary protocols including:

- Cisco's Tag Switching [4]
- IBM's Aggregate Route-based IP Switching (ARIS) [31]
- IP Switching (Ipsilon, now part of Nokia) [15]
- IP Navigator (Lucent/Ascend) [55]

These protocols, though differed in details, had the goal of producing simplified, and scalable IP networks each contributing in its own way to the evolution of MPLS. MPLS was set to address a number of shortcomings that were present in the current IP-paradigm. These included:

- A core transport network independent of the of type carried traffic
- Simplified high speed forwarding/switching of packets inside the core
- Provision of scalable networks
- Control over quality of service
- Traffic Engineering and the control of traffic routing

2.5.3 MPLS Architecture and Terminology

MPLS architecture defines two types of routers or switches: core routers, and edge routers. Core routers, also referred to as Label Switched Routers (LSR)s, are high-speed routers that reside in the core of an MPLS network, and participate in the establishment of LSPs using the appropriate label signaling protocol. The LSP is a configured connection between two LSRs in which label-switching techniques are used for packet forwarding.

Edge routers, also referred to as Label Edge Routers (LER) are routers that operate at the edge of the MPLS network. LERs serve as the gateway to connect between dissimilar networks (e.g. frame relay, ATM and Ethernet) and the MPLS core. They handle traffic from outbound networks and forward it on to the MPLS network after establishing LSPs, LERs plays a significant role in the assignment and removal of labels as traffic enters or exits an MPLS network.

2.5.3.1 MPLS Data Structures

MPLS routers maintain a number of data structures to assist in the interpretation and processing of labels. Packets that enter a LSR or a LER are checked against the Incoming Label Map (ILM) that consists of all incoming labels that an LER or an LSR can recognize.

The second data structure that MPLS defines is Next Hop Label Forwarding Entry (NHLFE). This table consists of all labels that can be pushed onto packets. This data structure contains label, outgoing interface, and next hop information [19][54].

The third data structure that MPLS routers maintain helps LERs decide which label to add to incoming IP packets. In MPLS, a Forwarding Equivalence Class (FEC) is defined as a way to group incoming packets into subsets of the total available traffic. In effect, it resembles a group of packets that are treated in the same by the router. Contrary to conventional IP networks, a packet in MPLS network is assigned to an FEC only at the entry time at the LER, whereas that is done at every hop in IP model [19].

Different forwarding treatments can then be applied to different FECs. FECs can have a generic meaning and can be defined based on several filtering policies (e.g layer 3 address, layer 2 address, transport protocol/port, etc.) Hence the third table that is maintained by MPLS LERs is the Forwarding to NHLFE (FTN) table. This table consists of all the defined FECs and maps the defined FECs into entries in the NHLFE table [19].

2.5.3.2 *Shim Header Definition*

The label defined by MPLS is integrated into a generic 4-byte "shim" header that is inserted between IP layer 3 header and layer 2 Datalink layer header. The shim header can further be tailored to fit special Datalink layers, such as Frame Relay, and ATM.

The label can be a generic number that uniquely identifies a set of data flows on a particular link. Labels only have "local" meaning, and are swapped at each hop change as packets traverse an LSP. Values for labels are agreed upon by neighboring LSRs before traffic starts to flow in. Most importantly, there must always be a one to-one mapping between FEC and label binding. That is, packets belonging to the same FEC essentially bear the same outgoing label [54].

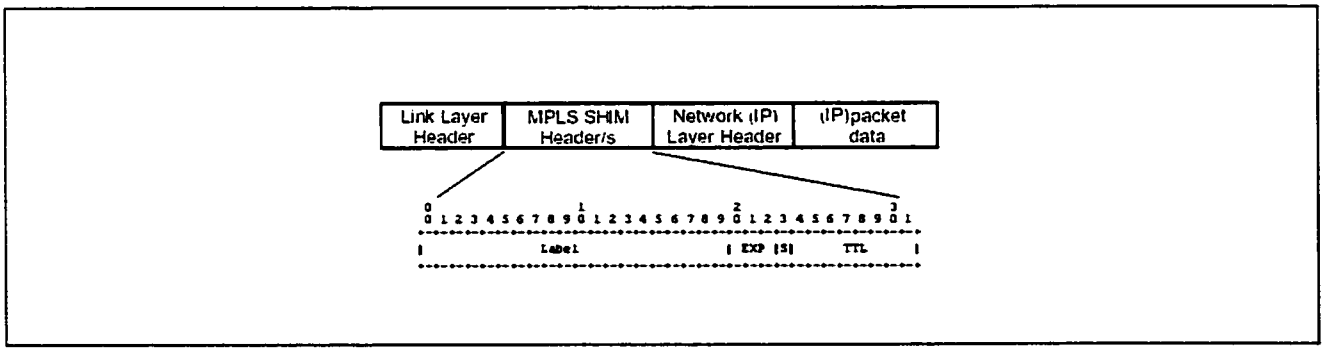

Figure 2.11 PPP and LAN MPLS generic shim header

The MPLS label consists of 20 bits. The label is used to determine how the packet will further be treated. The next 3 bits are labeled EXP bits. These bits, originally defined as experimental, are now standardized for the use of MPLS with Diffserv-defined classes. The next bit refers to as the "bottom of stack bit" (S bit). This bit is cleared when multiple shim headers are stacked — e.g. when traversing multiple MPLS domains, or defining tunnels. Hence, the S bit signifies whether what follows the shim header is the layer 3 header, e.g. IP header, or another stacked MPLS shim header. Finally, the shim header includes an eight-bit Time-to-Live (TTL) counter. This field is used to introduce layer 3 protection functions – e.g. traceroute, loop detection, and multicast operations -- to MPLS routers even though layer 3 IP-header is not available to the intermediate MPLS LSRs [19].

2.5.3.3 *MPLS over ATM and Frame Relay*

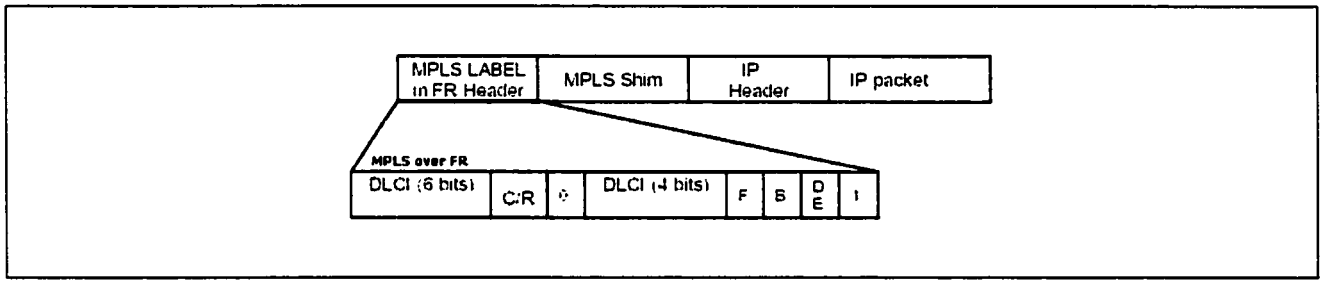

Figure 2.12 MPLS packet encapsulated over Frame Relay datalink layer

MPLS takes advantage of the fact that ATM is built upon a switching technology. However, The ATM LSR is constrained by the format imposed by existing ATM standards.

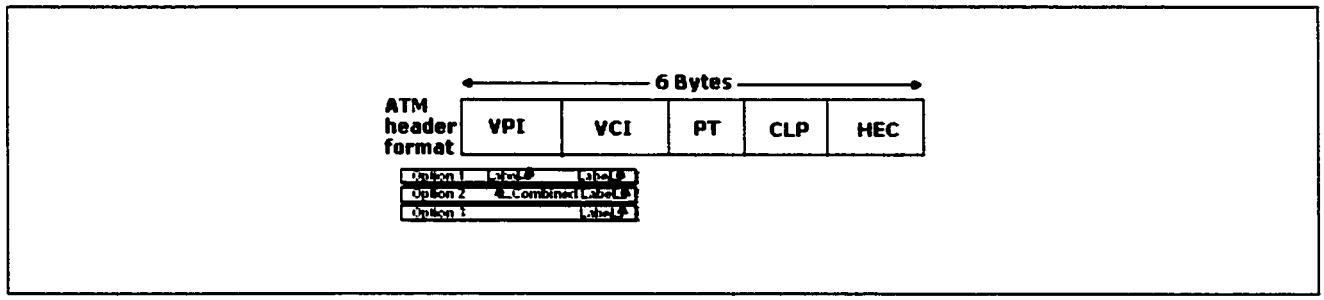

Figure 2.13 MPLS header over ATM

The MPLS forwarding label is copied into Virtual Path Identifier (VPI), Qirtual Channel Identifier (VCI) pair. However, a NULL shim is still retained to hold the EXP, the TTL, and S fields that would have existed if we were having normal MPLS forwarding. The null shim is added after

the ATM header in the first cell of the AAL5 frame. According to ATM standards, these fields will pass through the ATM network untouched. When the packet leaves the ATM network, and either goes back to layer 3, or moves on to general MPLS forwarding, these values will be available [53].

2.5.4 MPLS LSP Setup

Traditional IP distance vector unicast routing protocols (e.g. RIP) assigned static weight for paths based on the hop count that did not reflect the loading condition of the network. This made these protocols sufficient for connectivity purposes, while lacking traffic management capabilities. The introduction of link state protocols such as OSPF and IS-IS, helped in overcoming the scalability and convergence issues, but still did little in the way of traffic management.

MPLS allows greater efficiency in the transport over existing IP, ATM and Frame Relay networks, and provides certain QoS levels on "label switched" or fixed paths. IP networks suffer from delays in making routing decisions based on longest match lookup of the destination address from the routing table at each hop. MPLS, on the other hand, uses a short (20 bits) label carried in the MPLS shim header to forward packets, or relies on the native protocol (e.g. ATM, Frame Relay).

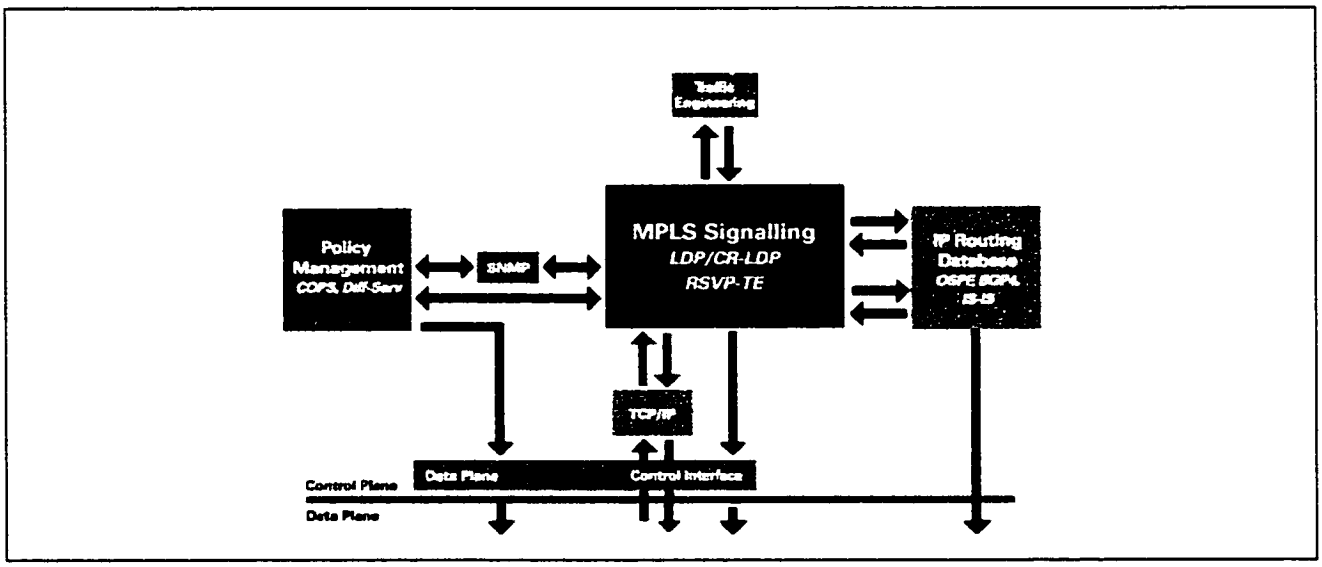

Figure 2.14 MPLS data and control plane

MPLS architecture is divided into two components: a forwarding component, and a control component. The latter is responsible to signal the LSP setup, maintenance, and label binding information. MPLS architecture defines a label distribution protocol as a set of procedures by which

LSRs agree on the meaning of labels used to forward between them along an LSP—in other words, an LSR informs another of the label/FEC bindings it has made. It is imperative to note that the LSP is set up before the start of data transmission, and is unidirectional (i.e. return traffic takes another LSP).

Once an MPLS enabled network is established, additional routing protocols (e.g. LDP, CR-LDP) [20], or extensions of existing routing protocols (e.g. RSVP-TE, BGP, etc) [18][35] can be used to activate the MPLS capabilities. LSPs can be established piggybacked based on the best effort paths defined by existing IGP routing protocols (e.g. OSPF, IS-IS, etc.), or by employing dedicated Label Distribution Protocol (e.g. LDP). This model of LSP setup is comparable to the IP hop-by-hop routing scheme that independently selects the next hop for a given FEC.

MPLS 's explicit routing is analogous to IP source routing, where the ingress LER specifies the nodes that constitute the ER-LSP. The path, specified here, need not be the shortest one. Moreover, resources may be reserved along this path with signaling protocols such as CR-LDP, and RSVP-TE.

2.5.5 Differentiated Services over MPLS

As described in the previous section, FECs are defined to aggregate incoming traffic from non-MPLS networks into classes at the edge boundary of a MPLS network to determine the forwarding path or LSP of different types of traffic. Hence, packets with the same FEC share the same LSP. However, to be able to allow multiple Ordered Aggregates (OAs) of different classes to share the same LSP with different priorities given to each, the Diffserv architecture was extended to the MPLS core. This approach eased the task of deploying end-to-end *"services"* with a single LSP, while allowing bandwidth borrowing between OAs of a customer [9].

Since the MPLS label is the only field that is checked by LSRs at every hop, the Experimental (EXP)– now renamed to CoS– bits in the MPLS shim header and part of the label field were adopted to map the IP defined DSCP field. When a single LSP carries multiple classes of service, the CoS bits are used to determine a specific queuing, scheduling, and drop policy. The EXP field consists of 3 bits that could only map to 8 classes referred to as E-LSP classes. In order to extend the mapping to include more Diffserv classes a new mechanism referred to as L-LSP PSC, in which the DSCP is mapped to a portion of the label field, was introduced [29].

In E-LSP, the EXP-to-PHB mapping can either be pre-configured, or explicitly signaled during E-LSP establishment. Hence, the LSR will determine the queuing and scheduling treatment (i.e. the PHB) to be applied on the incoming packet, by solely looking up the EXP field in the EXP-to-PHB mapping.

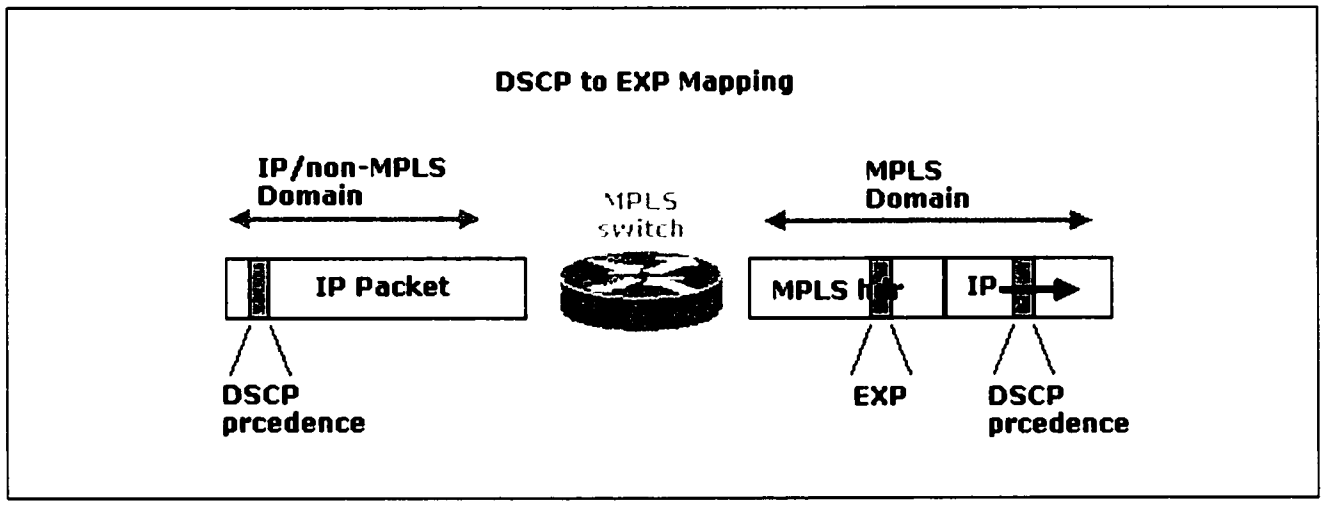

Figure 2.15 Mapping of IP DSCP to MPLS EXP bits

For L-LSP, the PHB is inferred from the combination of the label and the EXP in the MPLS header. The queuing and scheduling treatment is inferred from the label itself through a mapping function known to each LSR at LSP establishment time, and the drop precedence is carried in the MPLS EXP field, and is looked up from the EXP-to-PHB mapping at each LSR hop. Hence, the support of Diffserv in MPLS networks becomes possible by mapping Diffserv BAs into MPLS ER-LSPs [9].

2.6 CONCLUSION

From the above analysis, the following observations were drawn. First, the best effort service of the current Internet undoubtedly failed to provide any means of insurance for applications that otherwise would suffer from poor performance offered by best effort Internet service at congestion time.

The Intserv architecture succeeded to provide what the best effort service was not able to keep up with. However, the need of having RSVP-capable nodes to support end to end QoS dictated

replacing most of the Internet router infrastructure, which seemed a tedious and time-consuming job. In addition, the state information in Intserv increased proportionally with the number of flows, leading to a huge storage and processing overhead on routers, and causing serious scalability problems in its implementation in large ISP networks.

The Diffserv architecture seemed to carry some of benefits of RSVP [38], while overcoming problems of inscalability faced by the latter. To make this possible, the Diffserv architecture had to group micro-flows into more granular classes to be dealt with. Here, the amount of state information became proportional to the (limited) number of classes rather than the number of flows as in the case of Intserv. Yet, Diffserv alone failed to give hard deterministic guarantees on end-to-end service. To overcome this Diffserv can be augmented with some intelligent traffic engineering mechanisms to manage the network resources efficiently and effectively. Still for reason attributed with the IP routing protocols, traffic engineering in IP Diffserv networks was not an easy problem.

MPLS protocol introduced a solution to the traffic-engineering problem of IP networks. With MPLS it became possible to theoretically define the route that every packet will take. MPLS architecture brought along several advantages over the IP that can be contributed to: (i) its ease of implementation and maintenance of explicit LSPs that are not constrained by destination-based forwarding algorithms, (ii) its fast switching performance due to the short (20 bit) label "non-destination based" forwarding strategy, (iii) traffic trunks can be created and tunneled into a number of service classes associated with a set of attributes that define their behavioral characteristics, (iv) and finally leading to efficient implementation of the problem of traffic engineering in large networks.

This opened the doors for many improvements to be made. In future, MPLS in conjunction with Diffserv is expected to play a key role in providing QoS in WAN networks.

Chapter 3

TRAFFIC ENGINEERING IN MPLS DS-AWARE NETWORKS

Traffic Engineering is the process of efficiently distributing traffic flows across a network physical topology to facilitate efficient and reliable network operations. It is usually concerned with performance optimization of the underlying operating networks. In general, it incorporates technology and scientific practices into the measurement, modeling, characterization, and control of the network traffic. This knowledge is employed to achieve specific performance goals. In MPLS networks, measurement and control functions are key in the process of Traffic Engineering.

The demand for Traffic Engineering in the Internet mainly arises from the fact that current IGPs almost always employ the shortest paths in their computation of the best path to forward traffic. However, this leads to major downside when shortest paths from different sources overlap at some links, causing congestion. These links wind up getting over utilized, while longer paths between the same pair of nodes are under-utilized. This can be avoided to some extent by configuring a routing metric, however, it is very difficult to maintain metric consistency for all destinations.

Traditionally, ISPs base their services on an overlay model, where transmission facilities are managed solely by Layer 2 switching [17]. However, with MPLS, traffic-engineering capabilities are

integrated with layer 3 routing advantages, given the conditions imposed by the backbone capacity and network topology. Hence, it is possible to calculate routes through new algorithms based on certain required constraints. MPLS allows an LSP to be forcibly configured, so that all packets of a certain stream go through a specific router. This routing approach allows us to prevent traffic congestion over a specific route, since we can control what flows are to pass through the specific route. [17].

Some critics even argue that one day network capacity will become so cheap and abundant that these problems resulting from poor network utilization would be overlooked. However, it is our opinion that the bandwidth will remain a precious commodity for the few years to come. Even if the cost of bandwidth over wired links becomes eventually reduced, wireless bandwidth will always be limited, thus expensive. In our work, we assume that almost all ISPs are concerned with solving the above problems and wish to optimize their resource utilization by implementing efficient Traffic Engineering in their networks.

3.1 TRAFFIC ENGINEERING PERFORMANCE OBJECTIVES

The key performance objectives of traffic engineering can be classified as either:

1. Traffic oriented or,
2. Resource oriented.

Traffic oriented performance objectives carry aspects that improve and guarantee QoS of traffic streams. This includes ensuring certain bounds on the packet loss, delay, throughput, and enforcement of service level agreements.

Resource oriented performance objectives carry aspects that optimize resource utilization through efficient management of network resources. A prime concern is to ensure that some network resources (e.g. shortest best effort routes) do not become over-utilized and congested, while other available links along possible alternate paths are poorly utilized. This is a common case in the best effort model, where network resources are not fully and effectively exploited resulting in inferior performance [22].

Link congestion typically occurs when the available network resources are inadequate to put up with the offered load, or when the existing traffic streams are ineffectively placed onto the available resources; causing congestion on certain links of the network due to over-utilization, while others remain underutilized.

The first type of congestion can be addressed by: (i) acquiring more resources to increase capacity, or (ii) applying conventional congestion control techniques, or (iii) applying both [28].

Conventional congestion control techniques tend to regulate and re-shape, if needed, incoming traffic at the boundary of the ISP backbone in order to protect the core network nodes from being overwhelmed. These practices include: rate limiting, window flow control [13], router queue management, schedule-based control [28] etc.

The second type of congestion that results from the inefficient mapping of traffic onto available resources can effectively be addressed through Traffic Engineering. In general, adopting load-balancing policies can reduce this type of congestion. The objective of such strategies is to minimize maximum congestion or alternatively to minimize maximum resource utilization, through efficient resource allocation. Once congestion is reduced or avoided, packet loss and transit delay decrease, while aggregate throughput increases. Thereby, the perception of network service quality as is experienced by end-users becomes significantly enhanced.

Clearly, load balancing is an important network performance optimization policy. Nevertheless, the capabilities provided through Traffic Engineering should be flexible enough to allow network administrators to implement other policies that take into account the prevailing cost structure and the utility or revenue model [10].

3.2 TRAFFIC ENGINEERING IN MPLS NETWORKS

As MPLS emerges as the choice for the core multi-service network infrastructure, practically feasible and effective traffic-engineering functions are now possible through the easy implementation of Explicit Routes, or Constraint-based Routes (CBR) using CR-LDP [22], [23], or RSVP-TE [35] signaling protocols.

MPLS traffic engineering enables an MPLS backbone to emulate and expand upon the traffic engineering capabilities of Layer 2 ATM and Frame Relay networks. TE over MPLS is capable of routing traffic flows across a network based on the resources the traffic flow requires and the resources available in the network. In packet networks such as Internet, a simple static shortest path routing algorithm is employed-- e.g. based on hop count (RIP), or Dijkstra's link weight algorithm (OSPF). A packet is sent on the shortest path to its destination. If there are multiple shortest paths, the shortest path is chosen arbitrarily. In MPLS, with Constraint Based Routing it is possible to define the route of traffic flow as the shortest path that meets the flow QoS specification requirements (e.g. bandwidth, packet delay, loss, etc.). In MPLS-TE, the traffic flow has bandwidth requirements, media requirements, a priority versus other flows, and so on.

RFC 2702 [17] defines the concept of MPLS Traffic Trunk (TT) as an aggregation of traffic flows of the same class placed inside the same LSP. In effect, a traffic trunk signifies an abstract representation of traffic to which specific characteristics can be associated (see Figure 3.1). In this sense, TTs can be perceived as objects that can actually be routed (i.e. their path can be dynamically determined), much like emulating virtual circuits in ATM or Frame Relay networks. It is worth noting that, although it is common in practice to confuse TTs for LSPs, there is a fundamental distinction between a TT the LSP that it traverses.

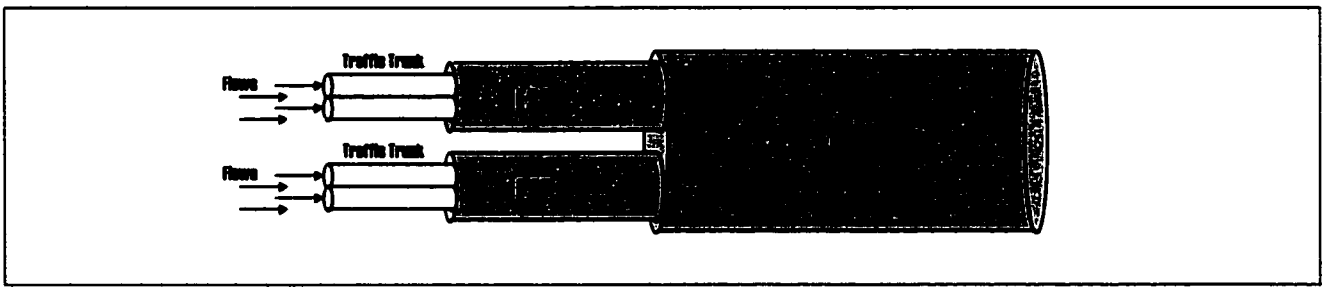

Figure 3.1 Partitioning of interface link into LSPs, Traffic Trunks, and Flows

The provisioning of a network backbone for intra-domain IP-traffic can be a big challenge, especially with the users' increasing demands and the mounting pressure on the service providers to provide certain QoS in terms of SLAs with customers, expressed in some form of loose guarantees on delay, loss, and throughput. This signifies the importance of traffic engineering, making an efficient use of the available network resources by offering the right path for the prevailing traffic. The need for dynamic adaptation is also necessary. An adaptive MPLS based traffic engineering

mechanism enables the backbone to be resilient to failures, even if many primary paths are pre-calculated off-line.

Hence, by imposing effective TE functions in a Diffserv MPLS network, prioritized traffic can be appropriately classified-- e.g. through utilization of the E-LSP Diffserv scheme-- and directed to pre-established LSPs to ensure better quantitative end-to-end QoS delivery even in an inter-domain environment. The mapping of traffic flows to a number of LSPs based on their QoS requirements can be done by dynamically assigning the flows to FECs corresponding to the required treatment.

3.3 TRAFFIC FILTERING AND DIFFERENTIATION AT THE MPLS EDGE

Path selection with some QoS guarantees has extensively been researched in both the ATM, and Internet world [21]. As mentioned earlier, careful network design and planning has to be constantly applied in order to accommodate network traffic growth and traffic distribution. Rapid Re-routing and Recovery (RRR) is crucial when problems arise, hence the importance of real-time optimization.

MPLS provides the capability to easily control the forwarding paths of subsets of traffic through a given topology. It is also possible to define explicit LSPs with certain bandwidth capacity, each LSP acting as a traffic-engineering tunnel. These tunnels can be assigned different costs corresponding to the grade of service offered on the LSP.

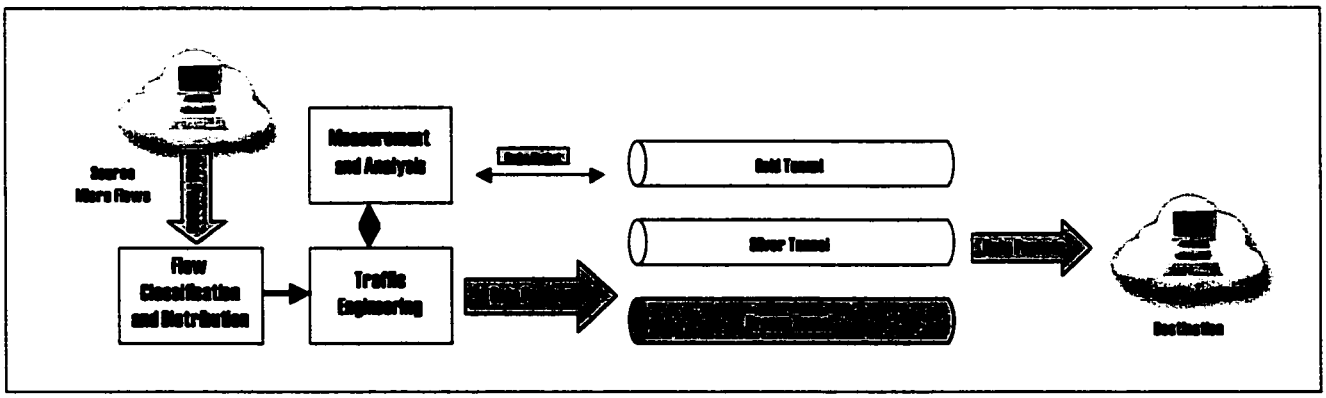

Figure 3.2 Core block functions of traffic partitioning on TE-tunnels with different grades of service

Figure 3.2 shows an example of three core differentiated traffic engineering LSP tunnels. The main functional blocks include: a classification subsystem that differentiates between incoming flows, a traffic partitioning subsystem that partitions the flows onto the available paths, and a measurement and analysis subsystem that carries out measurements and gives feedback on LSPs' performance to the traffic partitioning subsystem. The main intent of using multiple TE-tunnels of different classes of service is to meet specific traffic engineering, QoS, and protection requirements. In this case, prioritized traffic can be mapped to FECs bound to the corresponding tunnel that matches the QoS requirements.

3.3.1 Cheapest Path First (CPF) Alternate LSP Routing

Since Traffic behavior at the Ingress is generally unpredictable with varying incoming flow capacities to be mapped on established LSPs, the traffic loading on each of the established TE-tunnels will change with time, ranging from being lightly to heavily loaded, depending on the burstiness of the incoming traffic. This reflects on the level of quality these flows will experience when traversing the LSP. At the same time, it is possible to have other LSPs capable of delivering better quality assurances to time sensitive flows. Hence, a smart partitioning scheme to map premium traffic from a lower cost congested LSP to a less loaded, or higher-grade service LSP when needed, ensures the constraints defined in user are met at a minimum cost. In MPLS, it is possible to set up dedicated LSPs with reserved bandwidths for the purpose of distributing traffic across a given network according to given policies to achieve certain constraints.

To achieve this, the TE part of the architecture is responsible for dimensioning the network based on historical information and long time forecasts of the projected users' demands. This is typically associated with offline routing where load demands to be routed on all tunnels or LSPs are known a priori at routing time. However, in practice, new requests, or changes in the existing user's resource requirements over time will occur leading to the need to setup or reroute flows, or portions of it, based on the current state on each LSP. In our study, we assume that the LSPs are established and assigned a corresponding cost based on a long-term traffic matrix. The focus is on meeting QoS constraints at minimum service cost at the short-term due to fluctuations in the incoming traffic, or re-routing due to another LSP failure. To meet this we define an alternate LSP cheapest Path First

algorithm– we call (CPF)– that attempts to forward the time-critical traffic onto the cheaper path as long as it meets the QoS constraints of the time-critical

In a Diffserv network, the ingress traffic will consist of a collection of traffic aggregates or classes with varying QoS characteristics (e.g. EF, AF, BE, etc.). As mentioned earlier in Section 2.4, the AF class allows for some flexibility in terms of forwarding priority and drop precedence. Another special Diffserv group is the Expedited Forwarding group (EF) [26]. This class demands low loss, low latency, low jitter, and assured bandwidth end-to-end service. The Best Effort group identifies the nowadays-abundant Internet class of traffic for which no performance guarantees are provided by the network. In our study, for simplicity, we confine the discussion to two Diffserv classes, namely, the Expedited Forwarding class (EF), and Best Effort class.

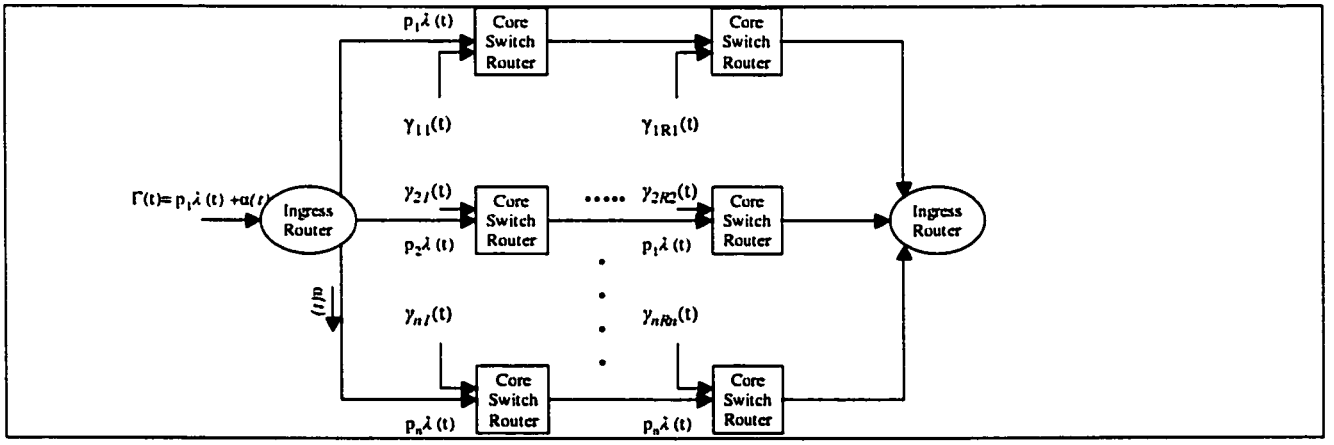

Figure 3.3 System model with traffic distribution

Assume the total incoming traffic capacity, Γ(t), is divided into two classes: EF with , and best effort. Traffic partitioning of the aggregates flows onto the established LSP paths can then be performed on: per-packet, per flow, or per class aggregate basis.

Partitioning on per-packet basis leads to the excessive overhead of packet reordering caused by packets traversing different paths. This is usually undesirable for some transport protocols (e.g. TCP). Rerouting on per-flow basis (characterized by a <source IP, source port, destination IP, destination port>) is capable of preserving packet order; however, it suffers from the scalability problems of tracking large number of micro-flows. A better partitioning scheme can be applied by shifting a fraction amount of the aggregate traffic present in the incoming queue at the ingress

router. This scheme also preserves packet sequence for each flow, while not necessitating any bookkeeping on a per-flow basis [12].

Let $\lambda(t)$, and $\alpha(t)$ be the time-dependent aggregate rate of arrival of EF, and BE class traffic at the ingress LER and destined for the egress LER. If we assume that $\alpha(t)$ is always routed onto a single LSP, we study the case of dynamically partitioning $\lambda(t)$ among the 'n' established parallel LSPs to achieve the desired SLS.

The traffic arrival rate of the EF class at each LSR's queue can be sub-divided into two parts: $\lambda_i(t) = p_i\lambda(t) + \gamma_{ij}(t)$, where $P = [p_1 \quad p_2 \quad .. \quad p_n]^T$, is the mapping factor vector of the total $\lambda(t)$ incoming traffic onto the corresponding LSP_i, and $\gamma_{ij}(t)$, $1 \le i \le n, 1 \le j \le R_i$ is the contribution of traffic from all other LSPs onto that node.

Please note that $\sum_{i=1}^{n} p_i = 1$.

Denote by λ_p the fraction of EF traffic $\lambda(t)$ that is shifted in a scheduling round along the EF TT on LSP L_p with a defined cost function C_p, the problem sums up to finding the minimal sum of products:

$$\text{Min } \{\sum_{p=1}^{n}(\lambda_p)C_p\} \qquad \textbf{Equation 3.1}$$

Hence, we ensure that λ_p is always routed across the lowest cost path that satisfies the SLS constraints at all times. This will effectively lower the client service cost while managing the network's resources ensuring prioritized paths. The choice of a cost function is important in making a selection decision when multiple alternatives exist. As mentioned earlier, the choice for the cost function, C_p, associated with a class TT on LSP_p, can be made by the management through negotiating relatively longtime contracts of users' traffic loads and reflecting the estimated utilization, or the longterm amount of the bandwidth used out of available bandwidth reserved for a certain class on an LSP tunnel [70]. For example, traffic levels are typically reported at 30-minute

intervals, and rate and utilization measurements over these intervals are used as a baseline for cost function assignment.

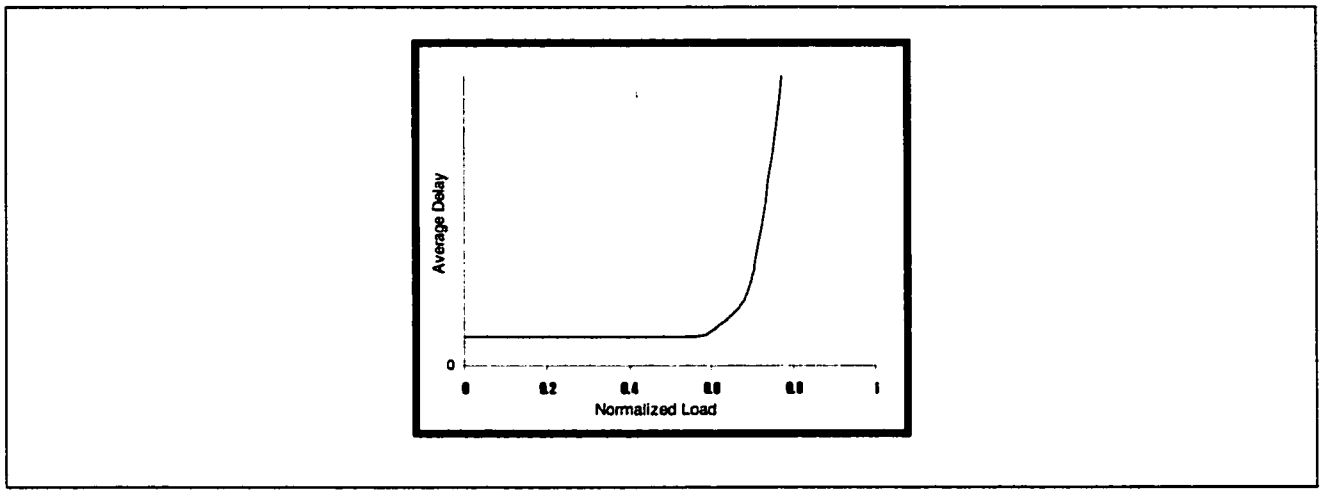

Figure 3.4 Average Delay versus normalized utilization plot

Assuming an LSP is modeled by a series of n (M/M/1) queues with service rate vector $M = [\mu_1, \mu_2, ..., \mu_n]^T$, where μ_i represents the service rate at node *"i"*, and arrival rate vector $\Lambda = [\lambda_1, \lambda_1, ..., \lambda_n]^T$, λ_i corresponds to the traffic arrival rate at node *"i"*, the average delay W experienced by a packet traversing the LSP is modeled as:

$W = \sum_{i=1}^{i=n} \frac{\rho_i}{(1-\rho_i)\lambda_i}$, where $\rho = \lambda / \mu$ is usually referred to as the utilization factor of the queue.

Hence, congestion occurs when the incoming traffic rate on a certain node approaches, or exceeds the queue service rate ($\lambda \rightarrow \mu$). When this occurs, queuing delays increase sharply beyond 80% load utilizations or higher--refer to Figure 3.4, and packets become increasingly susceptible to being dropped as buffer capacities on the link are exceeded [80].

Thus, to ensure premium service on certain LSPs (for example gold LSP), this LSP should be traffic engineered with a low traffic utilization factor, or with the provision of relatively high resource margins and possibly complemented with aggressive network monitoring. Consequently, it should be able to deliver a premium performance in terms of packet loss, and queuing delay. Thus, it is expected that this LSP would be associated with a high service cost. The silver LSP can be assigned a lower service cost than the gold LSP with an expected higher traffic utilization load than

the gold LSP. The bronze LSP is assigned the lowest cost, and does not provide any form of guarantees. It corresponds to a best effort like service, and at certain periods could be oversubscribed leading to poor QoS (an experience we are quite familiar with the existing Internet).

It is worth noting that the cost function C_p need not be a cost function per say, but rather a function that helps drive the network to a desirable operating point. For example, the network administrator might choose to leave certain paths lightly loaded so that they are used as backup, or premium paths.

Typically, performance parameters on an LSP or TT may include average packet delay, packet delay variance or jitter, packet loss rate, available bandwidth, etc. In our simulations and experiments, we employ an active measurement probe packet approach that is later detailed in Section 3.4 to constantly measure the expected packet delay on each of the established paths. The probe packet approach suggests a mechanism to calculate the end-to-end travel time of a packet by inter-communicating UDP packets along each of the established paths at regular intervals of time. It is also possible to measure the packet loss probability by encoding in the probe packet a sequence number to identify the number of probes transmitted, and another number that indicates how many packets received. This way, the user has a pre-knowledge of the QoS state on each LSP, and can dynamically map his traffic to the LSP that matches his QoS criteria.

```
for (; ; )
{
  sort LSPs(1 to n) in increasing order of Cp into array L;
  for (i=LSP1 to LSPn in L)
  {
        if (Dest(i)<Dth)
        {
              send (λp) on i;
        }
  }
  send (λp) on LSP(min(Dest));
}

Dest:   Estimated delay
Dth:    Threshold delay
```

Figure 3.5 Algorithm for Cheapest Path First (CPF) LSP satisfying delay constraints

3.4 PATH PERFORMANCE MEASUREMENT TECHNIQUES

Efficient traffic engineering techniques depend highly on the network traffic measurement process, and path state prediction over available paths between a source-destination pair. Network traffic measurements provide essential data for networking operation and research. As Internet grows in scale and complexity, the need for such measurements and their incorporation to advanced adaptive resource management algorithms becomes a necessity. The IP Performance Metric (IPPM) working group of the IETF [43] defines a set of standard metrics to measure the Internet performance. These metrics aims at evaluating the quality, performance, and reliability of Internet data delivery services.

3.4.1 Active and Passive Measurement Techniques

Passive and active measurements are two fundamental approaches used to track the performance and behavior of communication networks.

Passive measurement [72] is a traditional technique used to measure metrics pertaining to a certain network element by querying various traffic counters (e.g MIB counters) built into network devices. However from the application point of view, particularly real-time applications, end-to-end QoS metrics are of primary concern, and to obtain these measures at large scale from a large number of counters on routine basis, constitutes a difficult, data intensive, and time-consuming task.

Active measurement methods, on the other hand, are typically used to obtain end-to-end statistics such as latency, loss and route availability. This technique introduces controlled probe packets into the network, and measures the way the network handles those packets in terms of delay, path, and loss. Existing large-scale active measurement programs [45], [46], [79] have used probe traffic for this purpose at a coarse time scale (seconds, and sometimes minutes). On finer time scales, simple probe streams such as isolated streams with low periodic or Poisson distributed rates, have been employed to measure bottleneck bandwidth, available bandwidth, and investigate detailed statistical behavior of the delay, and loss [77], [78]. Other advantages of this technique include reducing the volume of measurement data needed as compared to the passive monitoring of high bandwidth links, and avoiding the need for certain privacy privileges to poll protected information on remote network devices. Yet, as in the case of almost all management-initiated traffic, the main

disadvantage of this technique is the amount of additional load introduced on network links and routers. To reduce the effect of the extra load introduced by test traffic, measurement projects typically use probe streams of average bandwidth lower than 10Kbps [76], the major challenge being able to provide more accurate projected future statistics without excessively increasing the test traffic's average bit rate.

3.4.1.1 Active Measurement Encapsulation Protocols

Active measurement packets can be encapsulated using existing protocols such as the Internet Control Message Protocol (ICMP) [56], the User Datagram Protocol (UDP) [67], the Transmission Control Protocol (TCP) [68], and IP Measurement Protocol (IPMP) [66].

While ICMP, TCP, and UDP protocols were not specifically designed to carry out this task, they are still being used in many network measurement tools that exist today for encapsulating probing packets; for example ping, fping, traceroute, mtrace, and the IP Performance Metrics (IPPM) group's One-way Delay Protocol (OWDP) [62]. On the contrary, the IPMP was introduced as an example of a protocol that specifically addresses the limitations of using existing protocols (ICMP, UDP, and TCP) to encapsulate packet probes.

Packet delay is a key QoS parameter for real-time applications and an important factor in making the decision to route some real time applications. Moreover, some transport protocols (e.g. TCP) rely heavily on the measured end-to-end delay to adapt their transmission rate to reflect network congestion state.

As mentioned earlier, the total delay experienced by a packet can be summarized into three components: propagation delay, forwarding delay, and queuing delay. The first two components are fixed, and can be seen as forming the minimum end-to-end delay possible. Delays that fall above this value are produced by the queuing delay, and their values are a good indication of the network's condition (e.g. if there network congestion).

3.4.1.2 Timing Control

There are several important considerations that need be taken into account in packet when using a packet delay measurements system. The sender and receiver have to agree on the protocol that will be adopted for the communication of the probe packets, including port numbers, packet

size, packet precedence, and encryption method. The second consideration is clock uncertainty that includes several factors itself.

Clock uncertainty is the result of four distinct notions: clock synchronization, accuracy, resolution, and skew. Clock synchronization is the measure to which two the clocks that participate in the measurement session agree on the value of time assigned at a certain instant. In real life, a clock might be running ahead or behind another. Offsets, in the range of ms between clocks running on telecommunications related units, are not uncommon. The accuracy is measured based on how the clock's time compares to the Universal Time Clock (UTC). The resolution of a clock is defined as the precision of a given clock. Skew measures the change of accuracy, or of synchronization with time.

Today, several techniques exist that enable clock synchronization with remote servers over the Internet (e.g. National Institute of Science and Technology (NIST) Internet Time Service (ITS)). The time information provided by the service is directly traceable to UTC (NIST). The service responds to time requests from any Internet client in several formats including the DAYTIME [32], TIME [33], and NTP [34] protocols. However to provide accurate synchronization of clocks independent of Internet introduced uncertainties, remote clients connect to Global Positioning Service (GPS), or CDMA based precise time sources. Nowadays, theses services are increasing in popularity and quality providing accurate time sources--either directly or through their proximity to NTP timeservers.

3.4.2 Active Measurements Using Conventional Protocols

3.4.2.1 Ping

"Ping" is one of the most frequently Internet tools used to measure path delays in IP networks. It is based on ICMP where a host or a router sends an ICMP Echo request message to a specified destination. Any IP device that receives an Echo Request, ideally, responds back to the original sender with an Echo Reply [56].

Most ping implementations measure the round-trip delay between the source and destination. Round-trip time is the interval between the sending of a datagram and the receipt of its acknowledgment. It implicitly measures both the network propagation delay, including time spent in

gateway queues, and any time spent at the receiver and sender processing the datagram and acknowledgment.

In Wide Area Networks (WANs), propagation delay is the most significant contributor to the round-trip time, however even on Local Area Networks (LAN), recent advances in reducing protocol processing times have made the propagation delay an important part of the total round-trip time.

The forward delay path is then deduced by halving the RTT measure. Often multiple requests are sent in a row, where the sender starts a timer that runs until the matching Echo Response arrives. Each of the attempts is measured, and the program output typically includes the minimum, maximum and average round-trip times. Echo Replies that do not reach the sender after a timeout elapses are also reported as lost indicating probably there exists no two-way path to the remote host, the remote host is down, or the packet was lost on the way.

A variation of this method to measure the one-way delay, is to construct an ICMP timestamp request packet that contains three timestamps: the originate time, the receive time, and the transmit time [56]. Assuming both hosts involved in the exchange of this packet have their clocks synchronized, the forward one-way delay then can be calculated as the difference between the originating and the receiving time. The reverse delay can be computed as the difference between the transmit time the packet arrives back to the originator.

3.4.2.2 Traceroute

Traceroute is an application that allows us to conduct measurement of individual link path delays that when summed up determine the overall path delay. A series of UDP packets are sent towards some destination, incrementing the TTL of the UDP packet by one each time they expire at every hop on the way to the destination. An ICMP error message type 11 "Time Exceeded" is sent back to the sender each time a router receives an expired UDP packet. This way, the routers along the path identify themselves by means of ICMP messages. At the same time, the sender is able to collect the round trip time at each hop.

3.4.2.3 *One-Way Delay Protocol*

The IP Performance Metrics (IPPM) [43] group has published a number of Request for Comment (RFC) documents that define frameworks for measuring the one-way packet delay and loss [43]. Several platform implementations exist that for the collection of these metrics SURVEYOR [46], RIPE [50], etc.

The OWDP [62] is capable of measuring packet delay in one direction as opposed to other techniques that measure the RTT. This is particularly significant since the path from a source to a destination may be different than the path from the destination back to the source (i.e. asymmetric paths, e.g. ADSL links). Round-trip measurements actually measure the performance of two distinct paths together. Measuring each path independently highlights the performance difference between the two paths that may traverse different ISPs, and even, different types of networks (for example, research versus commodity networks, or ATM versus packet-over-SONET).

In other cases, though the two paths might be symmetric (i.e. the packet travels the same physical path in both directions), they may have radically different performance characteristics due to asymmetric queuing. For example, in QoS-enabled networks provisioning network resources and admission policies in one direction may be radically different than provisioning in the reverse direction, and thus the QoS guarantees differ. Hence, measuring the paths independently allows greater flexibility, and accurate policy enforcing in both directions. Moreover, some application may rely on the performance in one direction, as the case in ftp application where TCP throughput is heavily determined by performance in the direction of data flow.

One key consideration that is to be taken in when implementing OWDP is router clock synchronization. One method to remedy the error introduced by clock synchronization in one-way delay measurements is to insist on echoing hosts using precise external time source such as those provided by the Global Positioning System (GPS), and the Code Division Multiple Access (CDMA) network. These precise external time sources result in a clock synchronized to real time with an offset of a few nanoseconds.

3.4.2.4 IPMP Protocol

The IP Measurement Protocol (IPMP) [66] was introduced as a protocol that specifically addresses the limitations of using existing protocols (ICMP, UDP, TCP) to encapsulate packet probes for network performance measurement process. For example, ICMP fails to deliver correct reports of delay to routers that protect themselves against Denial-of-Service (DOS) attacks by processing ICMP at a lower priority, or even discard ICMP echoes and serve only other kinds of traffic. IPMP provides the means to measure the one-way delay, taking path changes, while the probe packet is transit, into account. In this case tools like ping traceroute, etc. fail to report the precise values for the forwarding/reverse delay along a path. ISPs are often compared unfairly because large parts of the delay are outside their control. Internet surveys take too little account of the service actually provided by the ISPs they measure. Partly, this is because it is difficult. IPMP makes it possible for an ISP to discover and demonstrate the degree of delay introduced by it's network by deploying path record enabled routers at the boundaries of it's network.

IPMP supports measurement of the forward and reverse delay paths in a single packet, leading to more accurate RTTs, It also allows measurement of protocol based priority queuing, and the Bit Error Rate (BER) experienced along a path.

One advantage that IPMP carries over other techniques is that is does not require a router to maintain a real-time synchronized clock. Instead, the router that provides time stamp support can include whatever time stamp they have available at the interface in an IPMP packet. The relationship to real time is established through a separate packet exchange in which the router includes time stamps and the real time that they relate to. These can then be used at the measurement station to convert the received time stamps to real time.

This approach is intended to increase the implementation options at the router, thereby improving the accuracy of the final time stamp.

A typical implementation might be a free running clock (of known precision and maximum drift against real time) and a GPS providing regular time signals to the router through a serial port. The main CPU on the router receives the GPS's time signal via an interrupt and records the GPS time and the value of all interface clocks. This data, along with maximum measurement error

information is sent to the measurement hosts, on request, via the IPMP Information Request Packet.

3.5 SMOOTH RTT DELAY ESTIMATOR

The decision regarding the delay the packets experience while traveling through a path from the ingress LSR to the egress LSR should not be based on the value measured through a single or a few probe packets. An estimation algorithm, capable of "filtering out" the random variability, unavoidably affecting measured values, is needed. In addition, it should be able to project the estimate of the delay to the future, since the action will be taken after the probe packet has arrived and the estimate has been produced.

The ability to predict the expected delay is important, especially in a fast switching router, where decisions on where to route packets have to be made with the least delay possible. We propose an algorithm to estimate the current end-to-end delay based on the previous measurements, and the mean deviation.

The RFC793 algorithm for estimating the mean round trip time delay and available bandwidth is one of the simplest examples of a class of estimators called *recursive prediction error* or *stochastic gradient* algorithms [5][7][13]. Given a new measurement m of the RTT, the algorithm produces an estimate of the average RTT *'d'* by:

$$d \leftarrow (1-\xi) * d + \xi * m \qquad \textbf{Equation 3.2}$$

Here, ξ is a smoothing constant ($0 < \xi < 1$) that is related to the variance of 'm'. The formula can be rewritten as:

$$d \leftarrow d + \xi * (m - d) \qquad \textbf{Equation 3.3}$$

"d" represents the predicted value of the next measurement. The difference $(m - d)$ stands for the error in the current prediction produced from the old prediction plus some fraction of the prediction error. Hence, the error can be divided into the sum of two components:

$$d \leftarrow d + \xi * E_r + \xi * E_e \qquad \textbf{Equation 3.4}$$

where E_r is the random error introduced by the unpredictable fluctuations in the delay measurements, and E_e is the estimation error introduced by a bad choice of *'d'*. Hence, the $\xi * E_e$ term gives *'d'* a push in the right direction while the $\xi * E_r$ term gives it a push in a random direction. Eventually over a number of samples, the random error terms will cancel each other out and *'d'* tends to converge to the correct average [13].

It's obvious here that ξ represents a compromise, where large ξ increases the value of E_e while a small ξ minimizes the error introduced by E_r. ξ should be small enough to ensure that large deviations of m from the average value due to randomness will not force *"d"* from its actual value, while, on the other hand, preventing *"d"* from experiencing large fluctuations. In our experiments, we notice that the probe delay fluctuates sharply under loaded network conditions. Hence, to reflect a steady state of the link utilization, ξ should be given a small value (to prevent from allowing singular overshoots of delay samples from forcing the estimate of the average far from its actual value). Typical choices are in the range of (0.1 - 0.7) since smaller ξ's give a more stable a at the expense of taking longer time to converge to the true average [5].

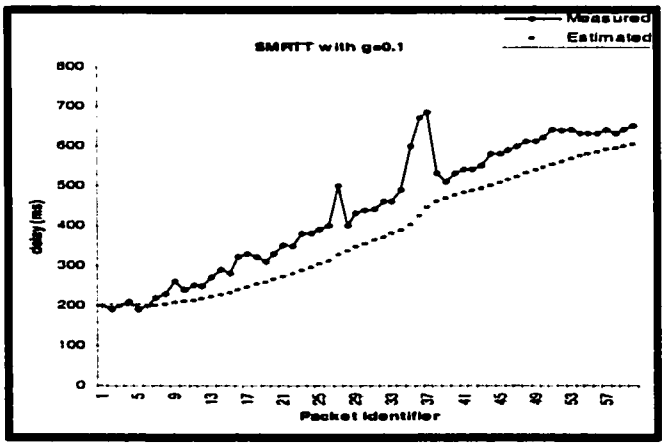

Figure 3.6 Estimated RTT with ξ =0.1

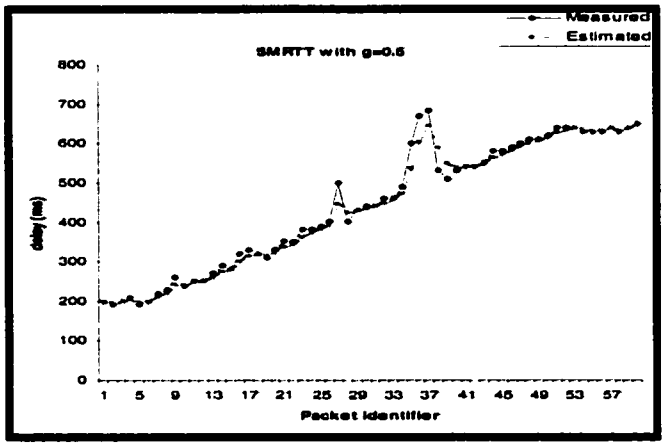

Figure 3.7 Estimated RTT with ξ =0.5

Figure 3.6, and Figure 3.7 show a predicted estimate of the RTT delay calculated according to the above formula for ξ =0.1, and 0.5. The RTT data was collected by pinging a remote host (www.yahoo.com) over the Internet with 60 probe packets of 1000 bytes each separated by 500ms each on a DSL connection. We can notice as ξ increases the estimated delay is able to follow sharper

changes in the delay. However, this also introduces higher values of predicted error. Hence, a moderate value of "ξ" has to be chosen in accordance with expected statistical behavior of traffic.

3.6 BOOT STRAP DELAY ESTIMATOR

Another technique for achieving an estimate of the delay from a set of previously sampled values is based on the application of the bootstrap estimator method, and is presented in [12]. The bootstrap method [71] was developed as a technique for assessing the accuracy of a parameter estimator, when other conventional techniques are not valid. The bootstrap estimation technique has been extensively employed in areas of signal and information processing, environmental, risk assessment, etc. studies.

Using this technique, it is possible to specify the number of observations needed to be taken to achieve a certain desirable confidence interval-- for example, that the estimated delay falls between two limits. The procedure can be summarized in a number of steps:

1. Draw a random m samples of the delay that produces the bootstrap: $Y= \{y_1, y_2, \ldots, y_m\}$
2. Calculate the mean for Y (say μ_1)
3. Repeat steps 1and 2 two a large number of times to obtain the means of n bootstraps: μ_1, $\mu_2, \mu_3, \ldots, \mu_n$.
4. Sort the means in increasing order: $\mu(1), \mu(2), \ldots, \mu(n)$
5. The desired $(1 - \alpha)100\%$ bootstrap confidence interval for the mean is $(\mu(q1), \mu(q2))$, where $q1 = (n\alpha/2)$ and $q2 = n - q1 + 1$

We can observe that this technique goes into a monitoring phase before being able to make an accurate decision on the delay estimate. This produces a sluggish effect before the estimate converges to the actual delay. Another observation about this technique is the overhead introduced in the process of producing the estimate.

3.7 CONCLUSION

In this chapter, we introduced an alternate LSP routing scheme we call Cheapest Path First (CPF). We combine the Diffserv technology with traffic engineering over MPLS to distribute delay-sensitive traffic across multiple established MPLS LSPs of different grades of service to meet delay constraints at minimal service cost. The user has the choice to specify an upper bound on the desired delay that the time-critical traffic can endure.

We propose an active probing approach to collect delay measurements along several parallel LSPs. We use the collected samples in a delay predictor that outputs a quick current estimate of the end-to-end delay. We carried out tests on a simulated prototype network to demonstrate the validity of our proposal.

The simulations in Appendix B demonstrate significant improvements for the Alternate LSP Routing (ALR) over the traditional shortest path routing algorithms that continuously forward traffic along the shortest path with no consideration to delay constraints. It also presented some economic improvements for the user by persistently attempting to send traffic starting from the cheapest service cost, and switching to higher cost LSPs only when needed.

Chapter 4

NETWORK TOPOLOGY AND EXPERIMENTAL WORK

4.1 DIFFERENTIATED SERVICES OVER LINUX

Linux offers a rich set of traffic control functions that make linux-based systems to be capable of emulating the functionality of most full fledged commercial routers. Current Linux kernels offer a variety of kernel-based processes, and user-space programs that offer the ability to associate queues, classes, and filters with devices. Linux kernel provides mechanisms required for supporting the architecture defined by IETF such as Intserv, and Diffserv..

4.1.1 Traffic Control

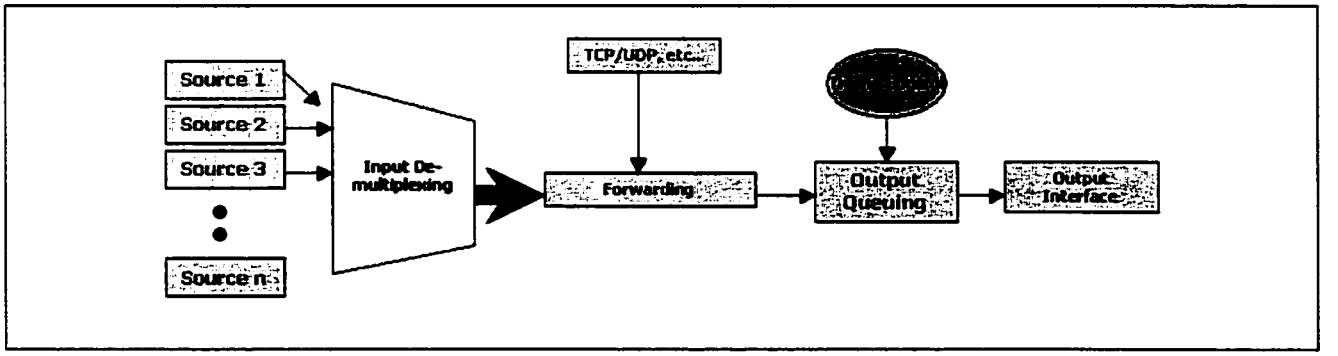

Figure 4.1 Processing of network data

Figure 4.1 shows the way received data is processed by the Linux kernel, and how packets are later forwarded to the network output device.

The forwarding mechanism, as mentioned in chapter 1, includes making a decision on the next-hop, the encapsulation type, etc. Packets are further queued with a certain queuing discipline, or transmitted right through a free output port. Traffic control, among other things, decides if packets be queued, or dropped (e.g. if the maximum queue size has been reached). Once traffic control releases a packet, the device driver picks it up and places it on the network. *'tc'* is made up of several components that interact in order to provide the desired functionality. These components are:

- Queuing disciplines
- Classes (within a queuing disciplines)
- Filters
- Policing

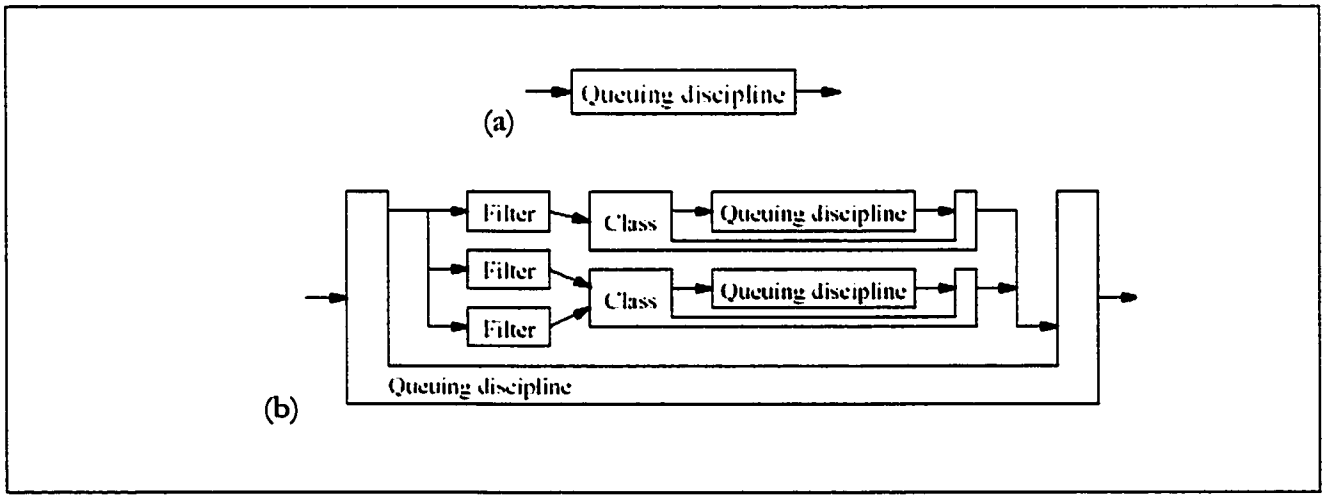

Figure 4.2 Network device interface with (a) basic FIFO queuing discipline, (b) Queuing with multiple classes in Class-Based Queuing.

Classes are attached to a Queuing Discipline (*qdisc*) and hold a portion of the total traffic. Classes and *qdiscs* are linked closely to each other. Typically, classes do not handle packets themselves; rather, have a *qdisc* associated with them perform this task (FIFO *qdisc* is the default). The splitting of traffic into different classes is done through the use of filters.

Figure 4.2(b) shows a more elaborate queuing discipline that can be associated with devices that provide special treatment for certain types of traffic. In this case, it is possible to define classes from within a queuing discipline and then associate the class with another queuing discipline (i.e.

nested classes). It is also possible to define filters to differentiate among classes of traffic, and then process each class at different priorities. Queuing disciplines in Linux can be arbitrarily chosen from a variety of available algorithms (e.g. Priority FIFO, Token Bucket Filter (TBF) [60], Random Early Detect (RED) [60], etc.)

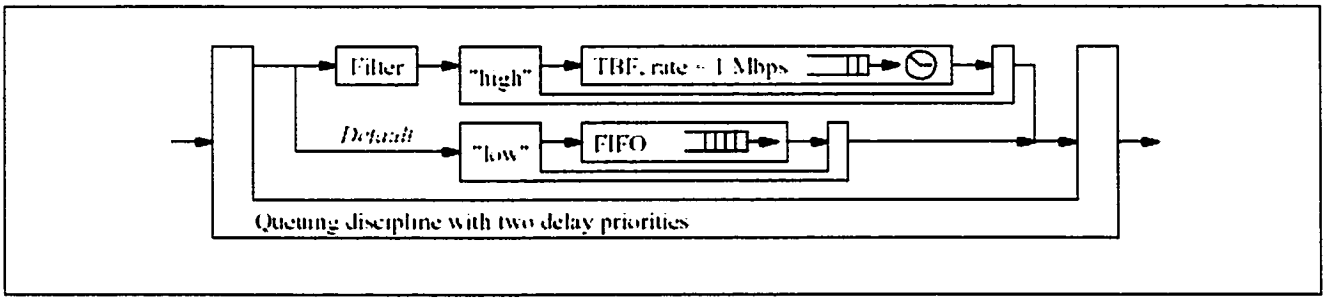

Figure 4.3 Using filters for low priority class (FIFO) qdisc, and high priority class (TBF) qdisc

Figure 4.3 shows a scenario for implementing two traffic treatments on a given output interface. Packets that match the filter's rule are directed to the high priority class and are treated using a TBF scheduling algorithm. All other packets that don't match the premium class filter are placed into the default low priority queue.

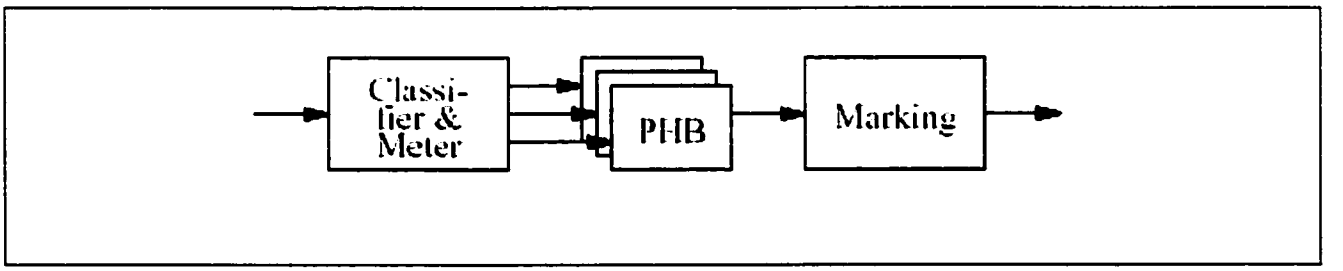

Figure 4.4 Building blocks of a DS-node

Filters, or classifiers in Linux (e.g. u32, ipchains, fw, etc.) are used by queuing disciplines to perform micro-flow classification. The dsmark queuing discipline is used to mark and retrieve the DS field. In addition, the tc_index classifier can be used to classify, and/or traffic condition packets. Further, Linux provides the capability to perform policing and remarking functionalities through the cbq and dsmark queuing discipline.

In Diffserv-enabled nodes, BAs are classified by retrieving the DSCP using dsmark qdisc, with the class held in the skb->tc_index. Nodes in multiple DS domains must also be able to distinguish

packets by the inbound interface in order to translate the DSCP to the correct PHB. This can be done using the route classifier, in combination with the ip rule command interface subset.

4.2 MPLS AND DIFFSERV OVER LINUX

MPLS for Linux is generally open-source projects to implement the MPLS stack, and portable versions of the signaling protocols (e.g. RSVP-TE, LDP, etc.) associated with MPLS, onto the Linux kernel. Several attempts currently exist to port MPLS to Linux (e.g. work supported by Ghent University, Information Technology Department (INTEC) [57] [58], and work supported by Fujitsu Laboratories Ltd. [95]).

MPLS for Linux [57] [58] initially started out as a protocol analyzer for the LDP protocol. It utilized a set of encode and decode functions developed by Nortel Networks. Currently, however, many parts of this project are still in the development stage, and with limited functionality.

The first task was to apply an MPLS patch, from [57], to our Linux kernel in order to enable the kernel to support the MPLS stack. We proceeded with installation on three other routers as shown in Figure 4.5.

Table 4-1 shows an example of the FTN (FEC to NHLFE) tables that were used on our testbed. This table is responsible for mapping incoming IP traffic of a specified FEC onto the correct output interface of the ingress router with the corresponding label. It is worth noting here that, in MPLS, the LSPs are defined unidirectional, requiring the establishment of separate LSPs in the opposite direction, when establishing two-way communication.

Table 4-1 Example of FTN tables at LER A, and C (see Figure 4.5).

MPLS LER	FEC	Label	Outgoing Interface	Next Hop
LER A	172.16.10.2/32	18	eth1	LSR B
	172.16.10.3/32	22	eth2	LSR D
LER C	172.16.40.2/32	16	eth1	LSR B
	172.16.40.3/32	20	eth0	LSR D

Table 4-2 Example of ILM tables at LSR B, and D (see Figure 4.5).

MPLS LSR	Incoming Label	Incoming Interface	Opcode	FEC
LSR B	16	eth1	Optional	172.16.40.2/32
	18	eth1	Optional	172.16.10.2/32
LSR D	20	eth1	Optional	172.16.40.3/32
	22	eth0	Optional	172.16.10.3/32

The Linux 2.4.17 kernel supports the Diffserv features described in the previous section. The next step was to enable the MPLS routers to be Diffserv capable. We applied a second patch to the same kernel, available at [57], in order to support Diffserv on our MPLS-testbed. We use the Class Based Queuing (CBQ) link-sharing hierarchy described in Section 4.1 to divide the output network device's bandwidth of each of the LSRs on the gold LSP among our two defined traffic aggregate classes EF, and BE-- sample scripts are listed in Appendix A.

4.2.1 EXP marking at the ingress LSR

As Ethernet frames arrive at the ingress router, they proceed from the Datalink layer to the IP layer. A function, in the kernel, is then called that takes the packet and inserts the shim-header between the IP header and the Ethernet header. The value of the label that the packet must hold is looked up from the Incoming Lookup Map (ILM) corresponding to the output interface that the packet must be sent on.

In the MPLS Diffserv E-LSP model, packets are forwarded according to the PSC referenced in the MPLS shim header's EXP field. It is also used to determine the drop precedence selected without any other information. It is possible to define a DSCP field to EXP mapping table that is maintained in the kernel. At the ingress router, the EXP bits are populated by the information looked up from this table. In Appendix A, we present the detailed steps used for marking incoming flows with certain DSCP, and then mapping the DSCP to the corresponding EXP value in the MPLS shim header at the ingress router.

On our testbed, we adopted the E-LSP model, and the EXP bits in the MPLS shim header are used identify the applicable PHB that the packet belongs to. The queuing disciplines to be used with each PHB are set using the 'tc' facility provided with the iproute2 package.

4.2.2 Forwarding at Core LSRs and LER

In the MPLS core, a mapping function is present that maps the EXP bits from the MPLS shim to DSCP by looking up form an EXP↔DSCP mapping table. IP packets arrive at the Ethernet layer and move directly to the MPLS layer to get forwarded based on the next hop. Here, the incoming label is swapped by looking up a new value from the NHLFE table, and is sent back right to the Ethernet layer. The root filter extracts the DSCP by mapping it to the EXP field in the MPLS shim header, and storing its value in skb→tc_index. The DSCP is then used to classify the packets into one of the defined queuing discipline classes. Appendix A includes the sample scripts used to define the PHBs associated with each Diffserv class.

4.3 NETWORK TOPOLOGY

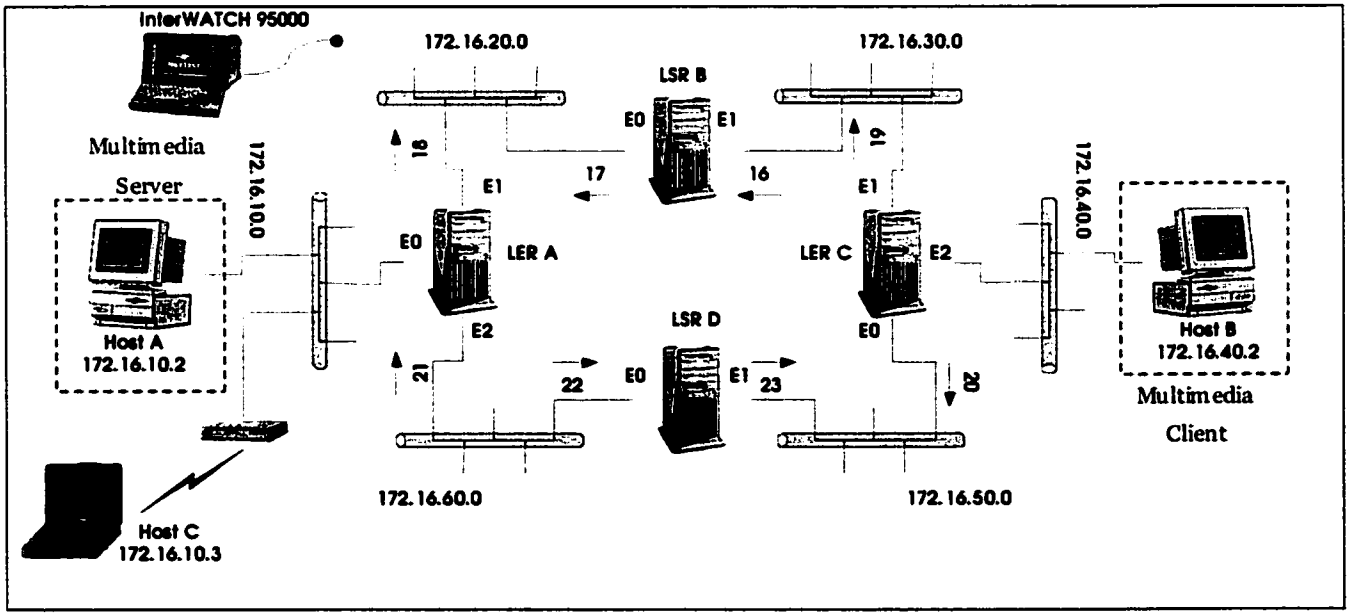

Figure 4.5 Testbed network topology

Our network prototype consists of an MPLS backbone network that connects two distant subnets 172.16.10.0/24, and 172.16.40.0/24. The backbone was formed of two edge routers (LER A, and LER C), and 2 core switch routers (LSR B, and LSR D). We divide our network to six separate subnets using a class B IP address (172.16.0.0) with subnet mask of /24. Two LSPs were established between LER A and LER C. We define LSP-2 (LER A - LSR B - LER C) as gold service and configure it to support differentiation of EF and BE classes' traffic at the edges and in the core,

and hence provide better QoS for premium users' traffic. LSP-1 (LER A, LSR D, LER C) was left out with best effort service to emulate a bronze grade service.

4.3.1 Routers and Clients Configurations

Our core test-bed consists of PC-based routers running Linux kernel 2.4.17, each containing one or more 100BaseT Ethernet NIC cards operating in full duplex mode. Each of the PC-routers consists of a 1 GHz AMD system with 60GB hard drives, and 512 MB of RAM. Routers were connected to each other through 100Mbps hub-switches.

The customer user network consists of video/audio server that services multiple remote clients' requests. The client-hosts have similar hardware configuration to the core Linux routers but run Microsoft Windows 2000 professional, and Advanced Server OS. Windows 2000 Advanced Server comes with Windows Multimedia Server support installed. The server is capable of streaming digital video and audio media with various formats.

We setup host "A" (172.16.10.2) as our video/audio server that services remote client's requests through our prototype network.

Incoming packets at LER A, originated from host A, and destined for host B are bound with a specific FEC and specified an outgoing label at LER C.

The network QoS behavior under several conditions was studied closely. Finally, our TE-algorithm was tested to ensure consistency with the simulation work.

To carry out accurate results on one-way delay, we designate host 'A' as our stratum 1 timeserver running the NTP service. All other routers in our test-bed will regularly try to synchronize their clocks to this timeserver.

4.4 PERFORMANCE TOOLS

To carry out accurate measurements on QoS parameters in our network, we employed a number of performance measurement tools, and standards. These included hardware-based

inter***WATCH*** 95000 protocol analyzers as well as some software-based packet-capture applications (e.g. Ethereal [47]. Iperf [44], tcpdump [87], and Netspec).

First, we used GN-Nettest's inter***WATCH*** 95000 Performance and Verification System. This platform provides a combination of modular interfaces and test-support applications for leading technologies including Voice over IP (VoIP), Multi Protocol Label Switching (GMPLS/MPLS), Universal Mobile Telecommunications System (UMTS), QoS measurement, ATM Signaling (UNI, PNNI), and IP Performance and Access technologies. The inter***WATCH*** 95000 can be connected simultaneously to multiple Ethernet segments, while monitoring network performance for router throughput and latency.

In order to capture packets at the core of our MPLS network, we make use of the Ethereal packet capture application [47]—an updated version of the well-known tcpdump application that runs on Windows and Linux and supports MPLS decodes. Ethereal is capable of capturing incoming and outgoing packets from a network device interface, while providing time stamp for the arrival/departure of each, and the inter-arrival time between consecutive packets.

The third tool used was Iperf, a java-based application developed by the National Laboratory for Applied Network Research (NLANAR) [45]. This application was capable of generating TCP, and UDP test streams with varying bit-rate and packet size for QoS measurement purposes.

4.5 TRAFFIC SOURCE S

One of the primary goals of traffic engineering is to be able to predict, with sufficient accuracy, the impact of the traffic generated by new applications on network resources, and evaluate if the required QoS is achieved.

To conduct performance study over new network mechanisms, or emerging new user applications, the characteristics of the traffic competing for the network resources have to be carefully considered. Several approaches exist to characterize and model traffic in computer networks; for example, it is possible to (a) analyze the measurements taken from actual aggregated traffic of an existing network, (b) focus on traces of traffic (e.g. traces from a given video or voice

application), and/or (c) concentrate on different classes of a certain application (e.g. in video applications: VBR, CBR, etc.).

Modeling and analysis of computer network traffic has been an area of extensive research over the past years [82][83][84][85], especially with the emergence of new Internet applications with varying resource and QoS requirements.

The use of realistic teletraffic sources in network performance analysis is absolutely essential for the accuracy of the reported results and the proper verification of new network technologies. For the purpose of performance evaluation, traffic generators producing traffic with statistics identical or very close to those of the actual sources can be used. Several source modeling approaches can be followed to model and reproduce the statistics of Internet traffic:

(a) *Trace-replay*: this method regenerates recorded streams of IP packets captured off LAN/WAN links through sniffer-like applications (e.g. tcpdump [87], etc.). This technique proves applicable in regenerating traffic of applications running over connectionless transport protocols (e.g. video/audio over UDP). However, it fails to reproduce accurate representation of network state when applied to traces of elastic sources (e.g running over TCP). This is due to the fact that these applications dynamically adapt to the existing network congestion state [86].

(b) *Aggregate traffic generators*: this method models traffic sources according to certain stochastic processes (e.g. Poisson, Markovian, constant bit rate etc.). However, this technique suffers from the same problem mentioned earlier in (a). Several open-source tools exist that adopt this approach (e.g MGEN [90], nemesis [88], Iperf [43], ttcp [87], etc.) -- refer to Figure 4.6.

(c) *User modeled behavior*: this method attempts to mimic the behavior of Internet users and underlying protocols and applications. This dictates a more detailed measurement and analysis of the various factors describing the model. However, this technique offers the advantages of flexibility, simplicity, and

provides physically interpretable model parameters. A number of works in this area have been presented in [82], [83], [84], and [86].

In our tests and performance analysis, we adopted the three types of aforementioned traffic sources, as well as real MPEG4-encoded [89] streamed video traffic.

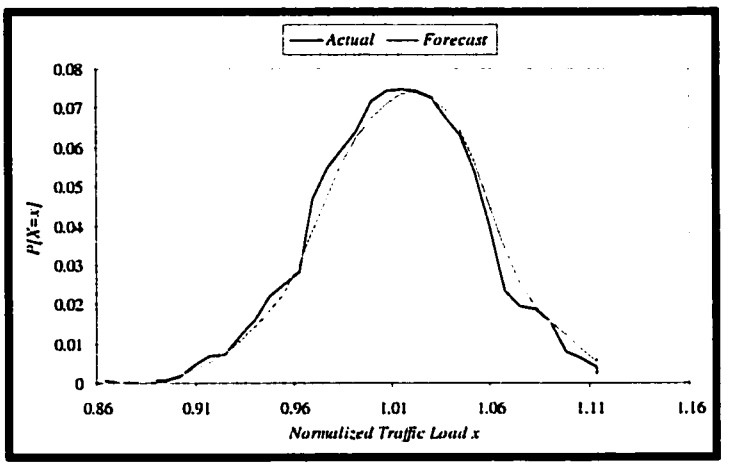

Figure 4.6 PDF plot of the normalized offered load generated by a Poisson aggregate traffic generator.

4.5.1 Synthetic Data Network Traffic

The World Wide Web (WWW) is a collection of documents and services available to the global Internet. Servers respond to clients' requests, delivering these documents. For example, the HTTP protocol [81] is a request-response protocol designed to transfer HTML files. Each transfer consists of a client request to a server for specific file(s); then, the server replies with the requested file.

To accurately evaluate performance under real-world Internet applications, the network behavior under WWW data traffic (e.g. http, ftp, telnet, etc.) should be considered. Originally, it might seem appropriate to regenerate real traces of captured TCP segments of variable size, and inter-arrival times. However, it should be noted that these quantities are influenced by the TCP flow and congestion control algorithms that depend on the latency and the effective bandwidth state on the path under study. Since this information is not known in advance, an accurate simulation of these types of data traffic has to take into consideration the network behavior as well as the actual

implementation of TCP algorithms. This approach was adopted in many works such as those reported in[82] and [83].

We adopted a similar approach to the one proposed in [81] in order to model web client behavior such as file sizes, and think times. We wrote multithreaded Java client/server applications that run at each end of a path or the link to be tested (refer to Appendix A). The client application launches TCP connections to the server and sends data through them in order to emulate the activity of a number of users running several web-based applications. The arrivals of new user sessions (clients) are modeled using a Poisson process with a mean arrival rate λ. This model was suggested in a number of works that were based on results extracted from actual measurements for a number of applications (e.g. Telnet, FTP, and WWW). Equivalently, the inter-arrival time of new sessions follows an exponential distribution with a Probability Density Function (PDF):

$$f(x) = \lambda e^{-\lambda x}$$

$$E(x) = 1/\lambda$$

The server's functionality consists of receiving clients' requests and responding with a number of requested files whose sizes are drawn from the "reply length distribution". Each reply is sent on the same TCP connection that carried the request. The number of data bursts (files) transmitted per session is selected randomly using the empirical data given in [83], and [84] with an average number

$$E(x) = 7.76$$

The size of each data burst can be modeled by a heavy tailed distribution such as Pareto distribution. The Pareto distribution has a Probability Density Function (PDF):

$$f(x) = \beta \alpha^{\beta} x^{-\beta-1}$$

$$E(x) = \frac{\alpha\beta}{-1+\beta}$$

where the location parameter α defines the minimum value x and the shape parameter β defines the heaviness of the tail. According to measurements reported in [85], β takes values from

0.9 to 1.4. The parameter α defines the mean file size, and hence the mean traffic load. In our application, we selected $\beta = 1.2$, and $\alpha = 2x10^6$. Between web page retrievals the user is generally considering the next action. This is referred to as think time. Based on [83] we assign to it an average value of 10 secs.

4.5.2 Synthetic Trace-Replay VBR Traffic

To experience the effect of mixing various sources of traffic present in real world networks, we incorporated in our tests a synthetic video server traffic source. The application was able to re-generate MPEG1/2/4, and H.263 synthetic traffic with variable/constant bit rate from encoded video trace files. The server was written in C code (refer to Appendix A) and was run over UDP. Our choice of C language was due to the superior performance in terms of time scheduling of the sleep function available in C.

Each trace file consisted of video frames $X = \{X_1, X_2, X_3, \ldots, X_n\}$. Each frame consists of a number of bytes, and belongs to a Group of Picture (GoP) pattern of IPB frames (e.g. generated by the MPEG encoding scheme). Frames are separated by time period dictated by the frame rate set at the time of encoding (e.g. a 25 frames per second translates to 1sec / 25 frames = 40 ms inter-frame time).

The X_i bytes that are read from the trace file are converted to UDP packets of maximum size of 1024 bytes (e.g. $n = int\{X_i/1024\}$, where $int\{y\}$ means smallest integer, greater or at least equal to y). The "n" UDP packets are then sent back-to-back at maximum speed before going into sleep state for the required time. In our tests we used trace files of "Star Wars", "Mr. Bean", and "Silence of Lambs" movies (available at [88], and [89]). A sample of video generated traffic is shown in Figure 4.7Figure 4.8.

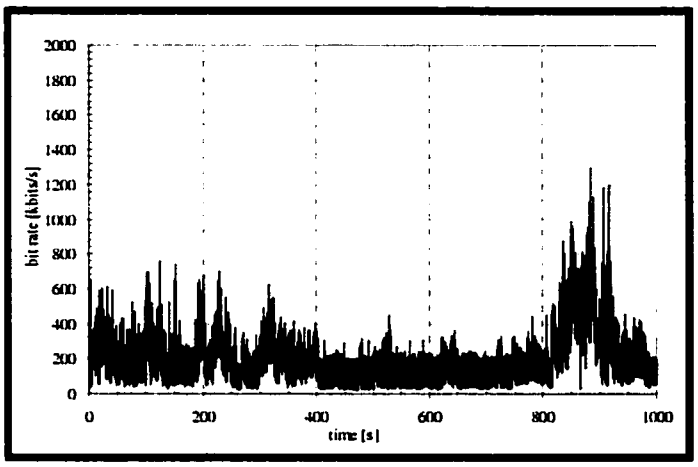

Figure 4.7 "Silence of The Lambs" bit rate trace (VBR MPEG-4 encoded, 25 fps)

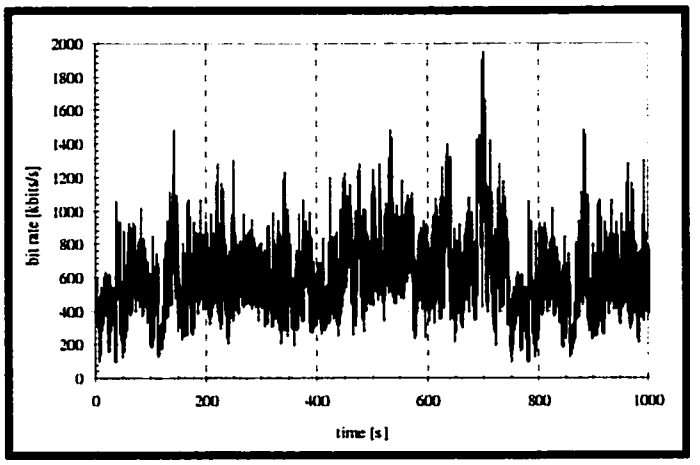

Figure 4.8 "Mr. Bean" bit rate trace (VBR MPEG-4 encoded, 25 fps)

4.5.3 Media Streaming Servers

To demonstrate the applicability of our network mechanism for providing better QoS guarantees to real-life applications, we integrated into our testbed two types of video/audio multimedia streaming servers (Windows Multimedia Server [92], and VideoLAN [91] multimedia server).

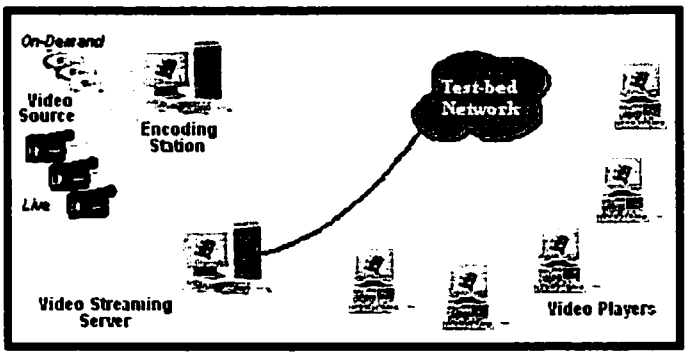

Figure 4.9 Typical video streaming network application.

The VideoLAN [91] client/server project is an open-source application that operates under several platforms including: Windows, Linux, and Mac OS. It can stream video from various sources (e.g. file, DVD, satellite and MPEG-2 encoder). It also includes a client (player), which can receive, decode and display MPEG 1 and MPEG 2 over UDP. VideoLAN server operates without any

control protocol (e.g Real Time Streaming Protocol (RTSP) [94], Microsoft Media™ Server (MMS) [92], etc.) for establishing and maintaining the connection between the server, and the client.

Windows Media™ Technologies uses MMS as its primary server protocol to send streams across the Internet, or LAN. MMS includes both Microsoft Media Server protocol/UDP (MMSU) and Microsoft Media Server protocol/TCP (MMST) that explicitly request the stream to use UDP or TCP respectively. MMS is Microsoft's equivalent to Real Networks RTP [27] protocol and uses a selection of commands for various jobs like connecting to the streaming server, requesting a file, acknowledging that the link is still in place as well as tasks like end of file indication, lost packet requests etc. MMS uses a special mechanism to monitor the effective bandwidth on the connection between the user and the server, adjusting the quality of video and audio accordingly. It can also be set up to compromise on different areas of the stream - for example preserving the quality of audio over video for a music video.

4.5.3.1 *Digital Data Encoding*

To provide digital content, we utilize a hardware encoder that is capable of converting analog video/audio signals into digital data that can be stored on digital media. We also perform tests on the client side with a software-based decoder. For encoding, we utilized a video capture encoder card supplied by Pinnacle Systems. The capture board generates AVI files of M-JPEG compressed video frames. Initially, we set the compression rate to minimum (2.5:1) so we achieve the best quality possible. This translates to a data rate of around 7.1MB/s for a capture resolution of 640x480x29.97 frames/second.

4.5.3.2 *Video Compression*

CODECs (compressor/decompressor) are tools that are used to help compress huge amount of video/audio digital data and reduce their transfer rates. There are two different types of CODECs: hardware CODECs and software CODECs. Hardware CODECs are usually more powerful, and equipped with fast DSP chips. Table 4-3 shows some of the current popular digital compression techniques.

In our experiments, we use the Windows Media Encoder (WME) [92] software to generate Class C ISO/MPEG-4 video sequences, and bbmpeg encoder [93] to encode the AVI captured video into MPEG-2 format.

Table 4-3 Comparison of popular compression techniques

	VCD	SVCD	DVD	AVI	DivX	ASF
Video Compression	MPEG1	MPEG2	MPEG2	Uncompressed	MPEG4	MPEG4
Audio Compression	MPEG1	MPEG1	MPEG2	Uncompressed	MP3/MPA	MPEG4
Size 1 hour video	600 MB	about 1 GB*	about 2 GB*	several GBs	about 350 MB*	about 120 MB*

* depends on the audio and video bit-rate.

4.5.3.3 *Live vs. Pre-encoded Video*

Networked video applications can be categorized into three main classes, (i) live interactive video (e.g., video conferencing), (ii) live non-interactive video (e.g., broadcasting), and (iii) playback of pre-encoded video. Each imposes certain delay constraints on the underlying network to ensure proper operation-- for example, a typical assumption in the case of interactive video conferencing applications is that a 150msec constraint on the roundtrip delay (from the time one of the users of the system says something, until the time this user receives an answer) exists.

Pre-encoded streams have frames available beforehand at transmission time; contrary to live broadcasts where video is captured compressed and encoded to generate frames "on-the-fly". Typically, delay constraints imposed by interactive applications are tighter than in the case of replayed media, where relatively small delays can be overcome by initially buffering data before playback begins. The degree of difficulty in meeting the constraints in each case depends on the bandwidth and delay characteristics of the underlying network.

4.5.3.4 *Sample Clips*

For our tests, we encode two video clips of 5-minute duration each. We use Windows Media Encoder Utility that accepts uncompressed AVI format as input and transfers it to WMV media format. We also set the maximum packet size to 1024 bytes during the encoding process to ensure

that packets will not exceed our backbone's set MTU (1500), and no additional delays will incur due to fragmentation. We encoded two sample clips with average traffic rates of 400Kbps, and 500Kbps. Figures 4.10 and 4.11 show the traffic rate distribution of the two sample clips, as captured by the protocol analyzer (IW95000) after streaming on an unloaded link.

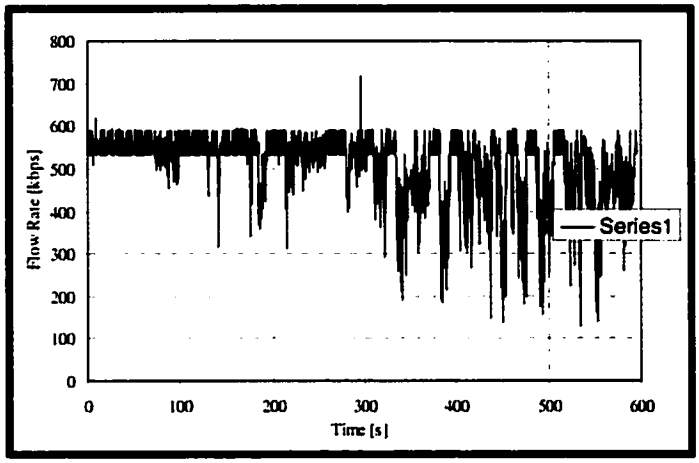

Figure 4.10 Video traffic flow rate: sample MPEG-4 video clip running at a rate of 30fps, producing an average rate of 500kbps.

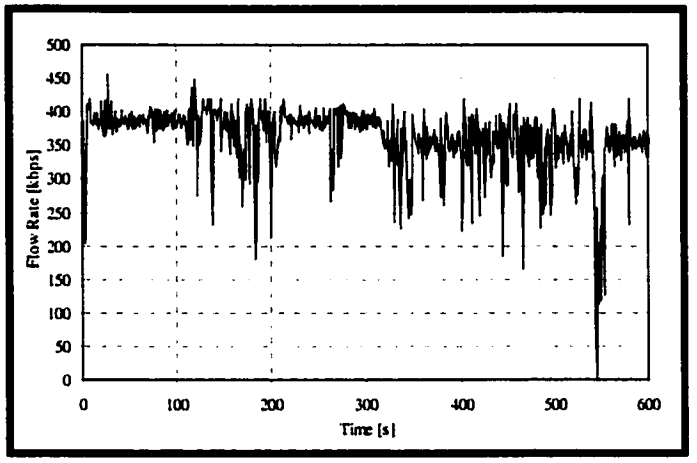

Figure 4.11 Video traffic flow rate: sample MPEG-4 video clip running at a rate of 30fps, producing an average traffic rate of 400kbps.

4.6 CONCLUSION

This chapter has presented a summary of the various experimental activities that were carried out on the prototype testbed. Section 4.1 described the different QoS features provided by the traffic control "tc" package supported by the Linux OS. Sections 4.2 showed the steps that were followed to patch the standard Linux kernel to enable MPLS and Diffserv features. Section 4.3 and 4.4 presented the setup configuration of the prototype testbed and the various performance tools that were used when conducting experiments on it. Finally, Section 4.5 described the different models adopted for generating the sources of traffic on the testbed.

Chapter 5

MEASUREMENTS AND RESULTS ANALYSIS

This chapter aims to present the tests carried out on our test-bed, and the results obtained. The goal is to evaluate the performance of our video stream in an MPLS-enabled backbone network with a heterogeneous mix of time-critical and non-delay sensitive traffic across different LSP configurations.

In order to evaluate our network performance in more realistic scenarios, we introduced into our testbed the traffic types, described earlier in Chapter 4, as "cross-traffic". These emulate the Internet background traffic, or traffic that "loads" the test-bed routers in order to emulate an operational environment. For example, the following is a set of applications, whose traffic profiles were reproduced by our traffic generators on the test-bed:

3. HTTP – asymmetric, bursty; fast response time; web browsing and HTTP
4. FTP download traffic
5. Streaming (real-time -VBR) – short delay and small jitter; streaming audio and video
6. Interactive (CBR) – stringent delay and jitter requirements, constant bandwidth; (e.g. VoIP, video conferencing, online chat, telnet).

As mentioned in Chapter 2, there are several components that, combined together, are able to provide certain aspects of QoS. One type of constraint an application may request is a bound on latency, in a form of $\Pr(D_i \leq D_{max}) \leq P_{min}$, where D_i is the delay for frame i, D_{max} is the maximum delay threshold (e.g. 200 ms), and P_{min} is the minimum success probability (e.g., 98 %). Additional

constraints include bounds on the throughput, jitter, and packet loss. The components that have been of particular interest to us in evaluating the network performance are delay, jitter and packet loss, experienced by a video reference stream under the background synthetic traffic models (e.g. aggregate WWW traffic, Poisson traffic, CBR traffic, video VBR traffic, etc.). In our results, we report the fractional bound of QoS in terms of Cumulative Distribution Functions (CDF). The experiments are conducted using the testbed shown in Figure 4.5.

By no means we claim that the results reported here have answered all the questions and uncovered all hidden potential problems. However, we claim with confidence that our work produced new results, explained successfully certain observed behavior, proposed new solutions. At the same time, it generated new questions that could lead to additional research work in the future.

5.1 PERFORMANCE UNDER BEST EFFORT SHORTEST PATH LSP

5.1.1 Setup and Methodology

In this case, we test the performance of our reference video stream on a best effort LSP connecting the multimedia server with the remote client host. All incoming traffic from network 172.16.10.0/24, having as destination the network 172.16.40.0/24, follows the route passing through LSP-1 (LER ***A***, LSR ***D***, LER ***C***) while the other path, passing through LSP-2 (LER ***A***, LSR ***B***, LER ***C***) is not utilized (refer to Figure 4.5). The RSVP-TE daemons are run on each of the LSRs, exchanging MPLS label messages for LSP setup. We configure a single best-effort LSP of maximum bandwidth capacity 10 Mbps (with a simple internal FIFO queuing discipline) on the outgoing links connecting ingress LER ***A*** to the egress LER ***C*** through LSR ***D***. All traffic types are transferred in this case through LSP-1, sharing the maximum allocated bandwidth.

The video stream is carried from server (Host A: IP 172.16.10.2) to client (Host B :IP 172.16.40.2) that runs a video client application. The background cross-traffic used for each experiment is detailed in Table 5-1. Background traffic, in this case, is generated through a separate Host C (IP: 172.16.10.3) injecting cross traffic to the remote subnet (172.16.40.0) across the MPLS network.

Table 5-1 Experiment scenarios

Experiment Number		Cross Traffic Type	Service Type	Transport Protocol	Number of Streams	Rate
Reference Stream	1,2,3,4	MPEG-2	Best Effort	UDP	1	2.8 Mbps Mean-bit-rate
Background Traffic	1	Aggregate constant-bit-rate (CBR)	Best Effort	UDP	Aggregate	10, 50, 100 % of tunnel capacity
	2	Aggregate Poisson distributed rate (3 simultaneous flows: packet sizes 1500/1024/128)	Best Effort	UDP	Aggregate	10, 50, 100 % of tunnel capacity
	3	WWW data Traffic (synthetic)	Best Effort	TCP	Aggregate	10, 50, 100 % of tunnel capacity
	4	Aggregate Video VBR MPEG-4 (synthetic)	Best Effort	UDP	10, 20, 25 flows of Mr. Bean and Star Wars movies (trace file)	400Kbps Mean-bit-rate

5.1.2 Results

In the experiments, we used the traffic generators described earlier in Section 4.5 to generate Constant-Bit-Rate (CBR), Variable-Bit-Rate (VBR), Poisson distributed, and WWW aggregated traffic sources (refer to Table 5.1).

We utilize the InterWatch 95000 protocol analyzer to accurately measure the one-way delay of every packet traversing the MPLS backbone network. To do this, the protocol analyzer monitors in promiscuous mode on the ingress subnet, and records arrival-time of each incoming packet sited at the ingress LER of our network. At the same time, the analyzer performs the same action at a second interface on the egress subnet for packets departing the MPLS network at our egress LER. The usage of the same clock (Protocol Analyzer's) to monitor arrival and departure time of every packet at the ingress and egress side of the MPLS network eliminates the error introduced due to clock synchronization. The approach was only used for collecting statistics about the reference video stream crossing our prototype network. While suitable for a laboratory environment, the method would not be applicable in a real-life scenario, where the ingress and egress POP reside at different

locations. In this case, synchronization of the ingress and egress routers through other means (e.g. use of GPS etc., for more information on clock synchronization please see Section **3.4**) has to be performed.

The difference between the recorded times is calculated, providing the one-way delay of the packets. It is also possible to determine the number of lost packets for a certain flow as the difference between the number packets sited at the ingress router and those sited at the egress router for a certain flow (refer to Figure 4.5).

Figures 5.1, 5.2, 5.3, and 5.4 show the one-way packet delay (OWD) cumulative distribution of the video reference stream under CBR, VBR, WWW, and Poisson distributed rate cross-traffic background loads across the best-effort LSP. For all background traffic types, more than 95% of the reference flow's total number of packets experience OWD higher than 10 ms. At heavy loads, the volume of competing traffic affects considerably the video traffic. At 70-100% of total LSP capacity, the OWD increases rapidly as the LSR's backlog of enqueued packets increases- 95%, 60%, and 50% of reference stream's packets experience OWD higher than 100-ms (severely degrading video quality) for VBR, Poisson, and CBR background traffic (samples of cross-traffic loads are shown in Figure 5.5, and Figure 5.6). Delays, in these cases, keep increasing until the router's queue is full, and additional packets start being dropped. Such large values of delay might render some of the packets useless, after arriving late at the destination.

Under full load WWW background traffic (Figure 5.3), the reference stream's packets experience lower OWD as compared to the other cases. This is due to the fact that WWW applications use the TCP protocol that adapts to the network congestion state, reducing the transmission rate in order to accommodate the available bandwidth. We also observe an increase in the OWD when introducing a mixture of VBR and WWW traffic as shown in Figure 5.3. This can be explained by the relative sluggishness of the TCP adaptation mechanism to the varying network condition under VBR loads.

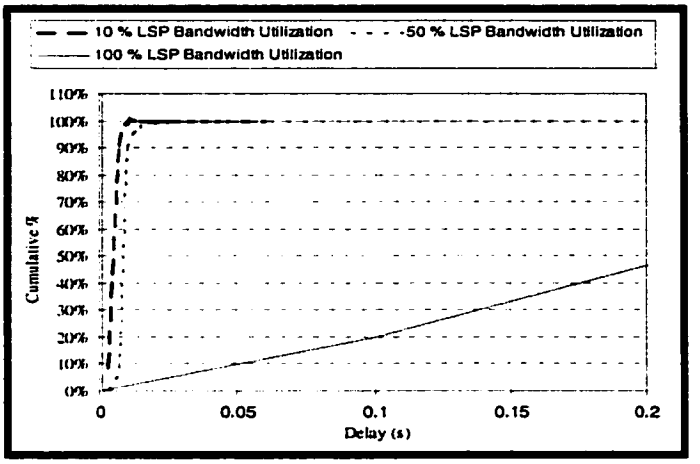

Figure 5.1 Percentage of reference stream's packets experiencing one-way packet delay across LSP-1 under a certain value, with 10, 50, 100% utilization of the LSP-1 capacity. [10% = 1 Mbps]. The background traffic source is of CBR.

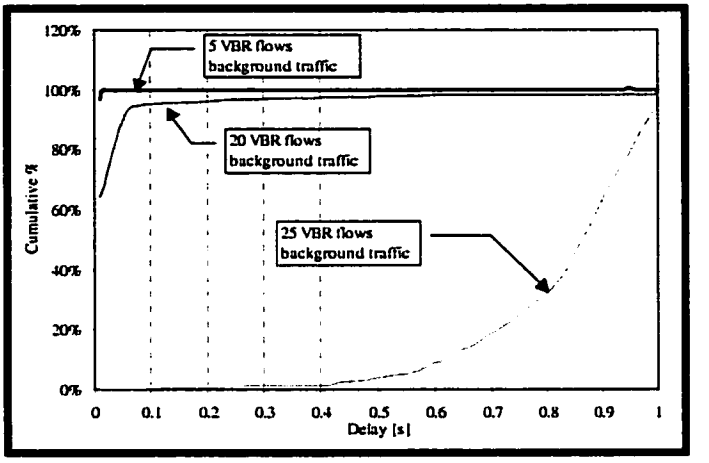

Figure 5.2 Percentage of reference stream's packets experiencing one-way packet delay across LSP-1 under a certain value, when 5, 20, and 25 MPEG-4 (variable-bit-rate) flows background traffic are sharing the link.

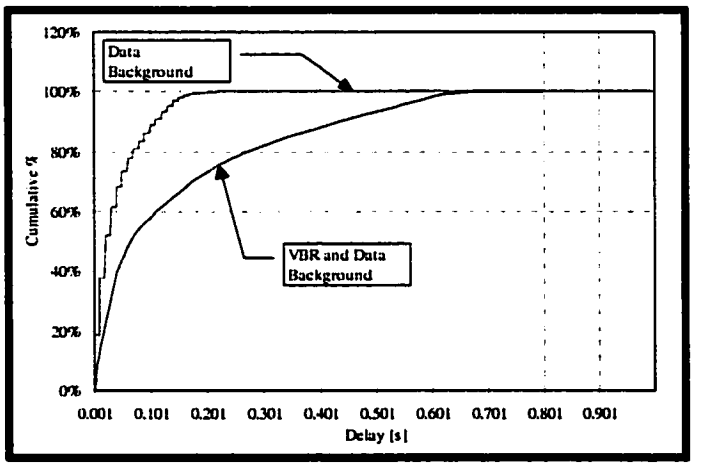

Figure 5.3 Percentage of reference stream's packets experiencing one-way packet delay across LSP-1 under a certain value, when: WWW traffic is sharing the link; ii) WWW and VBR video are sharing the link.

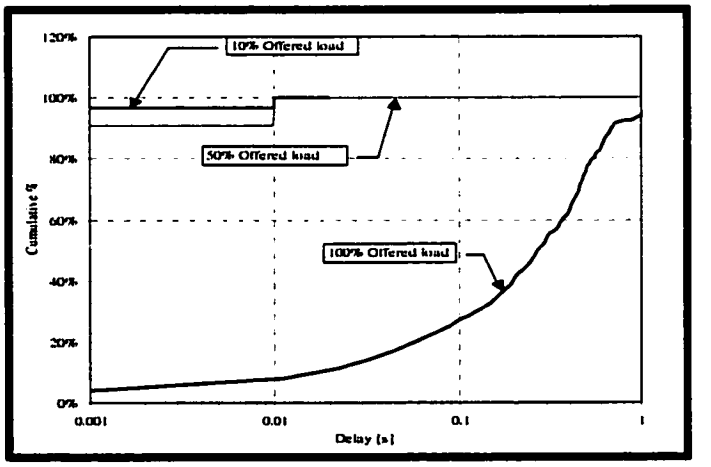

Figure 5.4 Percentage of reference stream's packets experiencing one-way packet delay across LSP-1 under a certain value, when: Poisson aggregate traffic is sharing the link.

The increase in the network delay affects the flow's throughput at the client side as well. The client's decoder expects frames to be "fed" sufficiently fast so that it can replay the stream at the specified frame rate (e.g. at 25 fps). With high latencies and relatively short replay buffer (e.g. in videoconferencing, or live video broadcasting) the decoder will starve out of data (decoder buffer

underflow) leading to frame skipping at the client at certain instances, or even termination of the connection due to a fatal loss of synchronization.

Figures 5.7, and 5.8.show the throughput measured at the video client's receiving end when streaming an MPEG-4 stream across the LSP-1 using Windows Media Server (WMS). As mentioned earlier, WMS uses the MMS protocol [92] as a control protocol to communicate messages between the streaming server and client in order to monitor the bandwidth, and request retransmission of lost –packets etc. We notice that WMS reacts to network congestion by first reducing the flow's bit rate (i.e. by reducing the video stream's frame rate) in order to minimize the effect. Eventually, as delay, loss, and throughput become unacceptable for transmission, WMS stops completely sending video packets, while continues sending the audio stream.

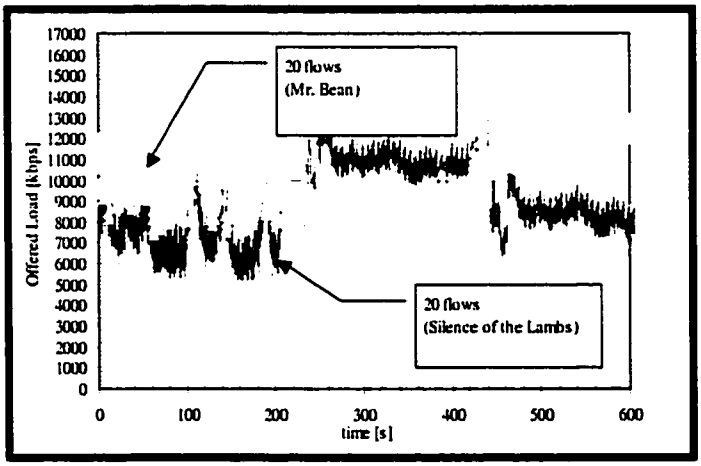

Figure 5.5 Offered aggregate MPEG traffic load (VBR) at the Ingress LSR on LSP-1

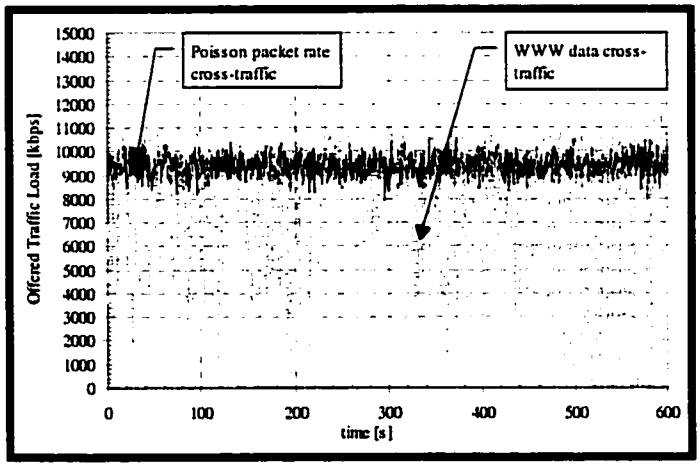

Figure 5.6 Offered aggregate traffic load: Poisson packet rate distribution, and WWW data traffic at the Ingress LSR on LSP-1

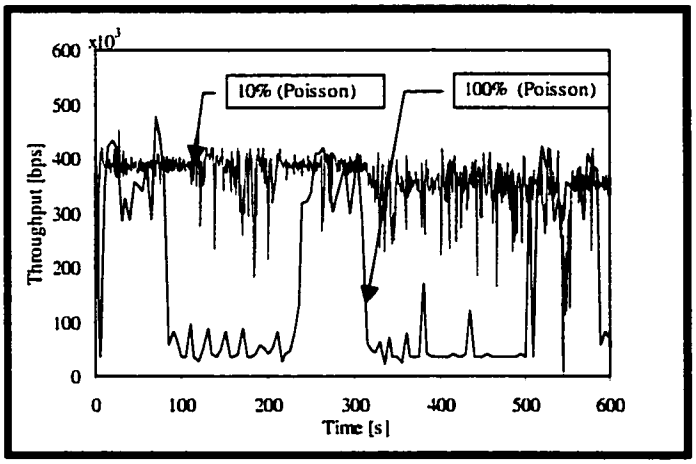

Figure 5.7 Reference stream throughput at client side (10, 100 % Poisson cross traffic): Windows Media Server with Target multiple-bit rate

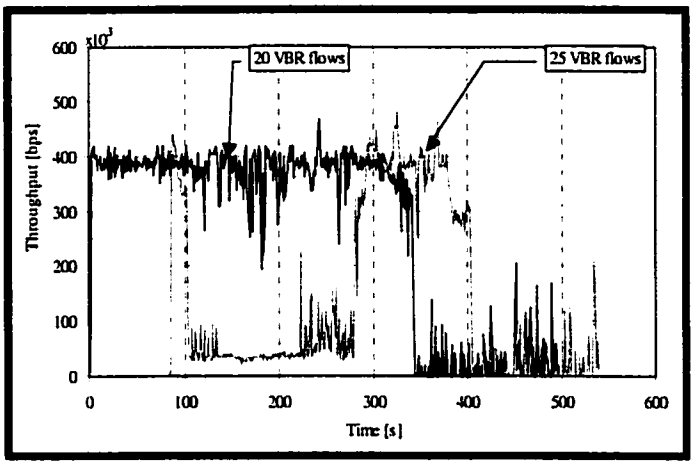

Figure 5.8 Reference stream throughput at client side (VBR cross traffic): Windows Media Server with target multiple-bit rate.

We calculated the Instantaneous Packet Delay Variation (IPDV), formally defined by the IPPM working group Draft [64] as the delay variation between consecutive packets, of the reference video stream experienced when the past mentioned background cross-traffic types are applied (Table 5-2).

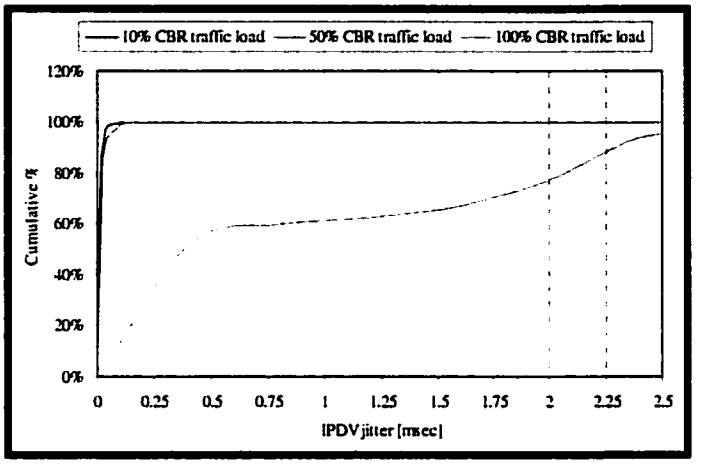

Figure 5.9 IPDV of referenced video stream packets under 10, 50, 100% load of CBR traffic.

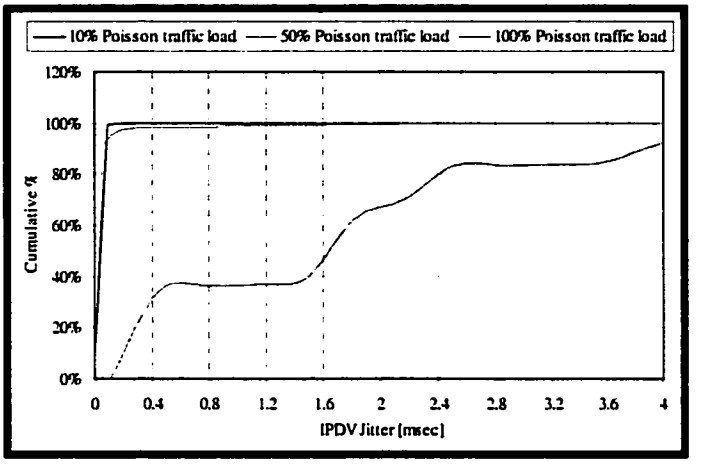

Figure 5.10 IPDV of consecutive packet of video reference stream under 10, 50, 100% load of Poisson rate traffic (3 simultaneous flows 128, 1024, 1500 bytes packet sizes).

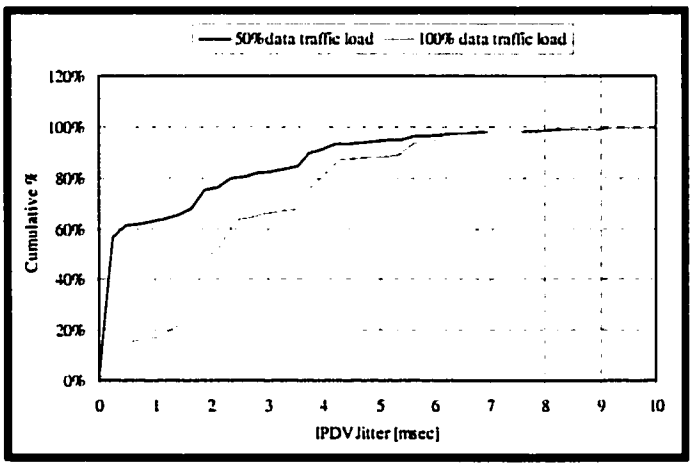

Figure 5.11 IPDV of referenced video stream packets under 50, 100% load of WWW data traffic.

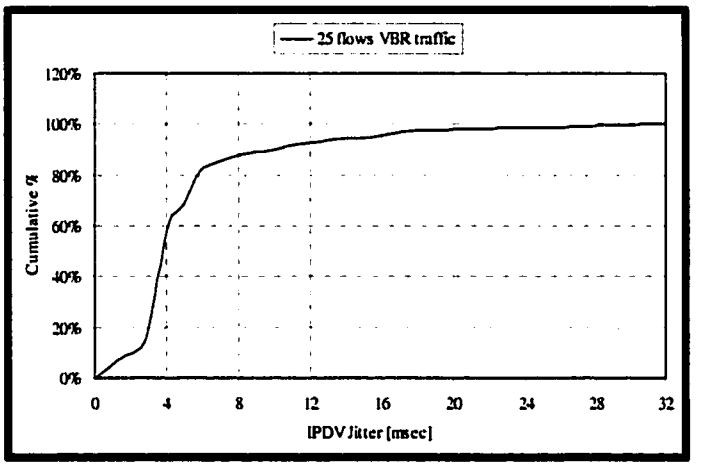

Figure 5.12 IPDV of referenced video stream packets under 25 flows of MPEG-4 (synthetic) VBR streams.

Jitter, sometimes referred to as "arrival distortion", results from the variation in the queuing delay experienced by consecutive packets of a flow. Jitter adversely affects video playback quality across a network. It causes video data packets from the same frame to arrive at the receiving end at different instants (not within one frame display time). Typically, with high jitter bounds (e.g. Pr (Jitter>20ms) > 10%), the video playback often appears "jerky." The jitter's effect on the perceived quality can usually be alleviated by buffering data at the receiver end (i.e. adding an artificial delay). However, this additional buffering increases the end-to-end delay. For interactive video based

applications (e.g. video conferencing, or other Distributed Interactive Virtual Environments applications) these long buffering delays are unacceptable for retaining interactivity.

Figure 5.11 shows the IPDV for different CBR background traffic utilization. As expected, under CBR traffic load, packets tend to experience comparable delays resulting in low IPDV. In this case, at 100% load almost 90% of the video stream packets have an IPDV lower than 2 msec. We noticed slightly higher values for IPDV with Poisson distributed background traffic (almost 90% of the video packets had IPDV lower than 3 msec). This is explained by the variation of the traffic arrival rate at the routers' queues that translates into varying queuing delay per packet. Under WWW data traffic, we notice higher values of IPDV (almost 10% of the packets experience IPDV that is higher than 7 msec). We also notice a considerable increase in IPDV when mixing the video stream with aggregate VBR background traffic (more than 10% of the video packets experience IPVD greater than 16 ms). As expected, the abrupt traffic fluctuations cause delay fluctuations, resulting in increased values of IPDV.

We also conducted a number of experiments to determine the effect the variation of the per-hop buffer size has on the packet loss rate of our reference stream, under the different cross traffic types. Packet loss degrades video and audio quality; hence, assigning large buffer size to preserve all packets is desirable. However, depending on the application type, packets are assigned a maximum bound on the allowable end-to-end delay (e.g. 150-ms for interactive video applications) before being considered lost by the application, and subsequently, become of no significance [61] (for example, in video applications, the decoder can not wait for late packets in order to display the current frame). Figure 5.13 shows the increase in the average one-way delay when increasing the router per hop buffer size from 3packets to 3000packets on each of the routers on the path. With the total aggregate arrival bit rate reaching volumes higher than the LSP's capacity (or service rate), the queue size of backlogged packet grows bigger, with packets coming later in time facing even higher delays (delay reaches values higher than 1s under high buffer size for CBR, and Poisson rate traffic). When mixing WWW data traffic with the video stream, we noticed that the delay tends to settle to a value close to 150 ms after certain size of the buffer. This can be attributed to the adaptive nature of the TCP data traffic that reduces the transmission rate to share the bandwidth with the video stream. In Figure 5.14 we show the loss rate experienced at the "worst-case" scenario (full load background traffic) for the different types of traffic.

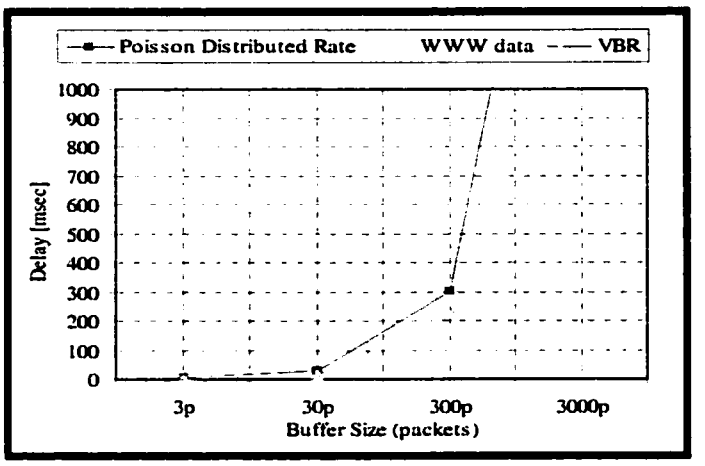

Figure 5.13 Average one-way delay versus router buffer size in packets. (100% background traffic utilization)

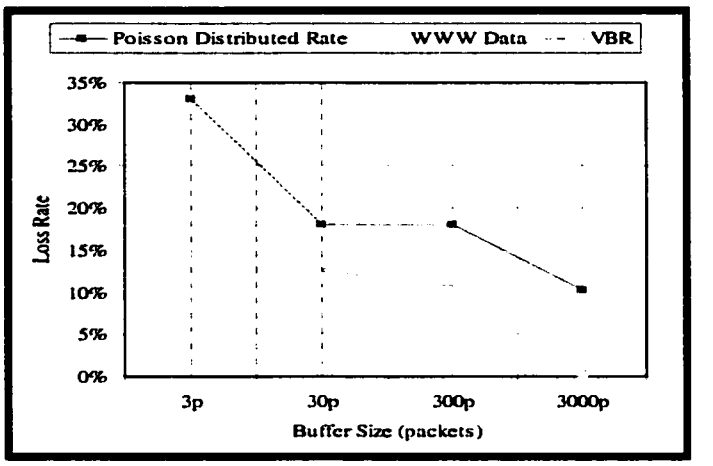

Figure 5.14 Video stream loss rate versus router buffer size in packets. (100% background traffic utilization)

5.2 PERFORMANCE ON DIFFSERV-ENABLED LSP

5.2.1 Setup and Methodology

As mentioned in chapter 2, the Diffserv architecture defines a set of technologies that allow the network to offer different levels of network QoS to different customers and their traffic streams. Traffic streams can be aggregated to a small number of behavior aggregates that are each forwarded using the same PHB (e.g. EF, AF, BE, etc.) at the router, thereby simplifying the processing and the associated storage.

In this case, we explicitly setup LSP-2 (LER A, LSR B, LER C) connecting the remote subnets (172.16.40.0 and 172.16.10.0) by running the RSVP-TE daemons to exchange MPLS label messages. The total capacity of each of the links on LSP-2 in this case is configured to 10Mbps. Two classes/Traffic Trunks (TTs) are setup: a high-priority (marked with EF DSCP), and a low-priority TT (marked with BE DSCP). Several different traffic-forwarding algorithms were described in Chapter 4. We test the performance of the EF stream under two types of schedulers: (1) Linux Priority-FIFO (P-FIFO), and (2) Class Based Queuing (CBQ).

P-FIFO is a simple, easy to implement algorithm, enabling the development of low cost networking products. CBQ, on the other hand, enables the network management to perform resource reservation (e.g. a class can be allocated a certain bandwidth resource). Scheduling of

packets is then made by "running" computing algorithms that determine the service the classes have received, and safeguard the fairness of the resource sharing process. For example, if a "non-isolated" class does not produce traffic volumes at the level of its resource allocation, another "unbounded" sibling class producing traffic a rate higher than its allocation (i.e. over-limit) can consume the unused portion of resource. Different policies can be applied to regulate this sharing process.

The flows used in our tests are generated by the hosts belonging to the IP subnet (172.16.10.0) on the MPLS network's ingress side and are sunk by the hosts of the subnet (172.16.40.0) at the MPLS network egress side. The offered traffic load from remote network (A) to remote Network (B) across this LSP-2 is divided into two types: a time-critical traffic (marked with EF DSCP), and non-time-critical traffic (marked with the BE DSCP). The VideoLAN streaming server generates MPEG-2 flow (2.8 Mbps mean bit-rate, and 3Mbps peak rate) of UDP packets marked with EF DSCP. Marking of incoming packets is based on multi-field classification (using Linux iptables firewall) applied to traffic in the input direction at the ingress router. In our experiments it is performed according to the content of the source and destination IP address fields, and the transport protocol source/destination port in use. For the traffic sent to BE class we use UDP traffic of CBR nature in order to reduce the number of variables affecting the system, and be able to better identify the mechanisms that affect performance.

As detailed in Chapter 4, a mapping function exists at the MPLS ingress router that maps the IP-DSCP (e.g. EF = 0x2e) to the corresponding EXP (e.g. EF/EXP =001) so it can be inspected by the core MPLS routers.

5.2.2 Priority-FIFO Based Differentiated Services:

For P-FIFO scheduling, two queues are defined at each node on the DS-enabled LSP: one for EF traffic, the other for BE traffic. As mentioned earlier, video is classified as premium service and is placed in the EF buffer, whereas background traffic is placed in the best effort buffer. We collect results on QoS parameters for the EF stream flow with different volumes of background traffic loading.

The EF one-way delay under the Priority-FIFO scheduler proved to be affected by the best effort packet size. This is due to the fact that an EF packet arriving at the node while a BE packet is in the process of being served, has to wait.

We experimented by changing the BE packet size while keeping the background traffic rate at maximum link capacity (10Mbps) and noting the effect on the video (EF) packet performance. In order to estimate parameters, we repeated every experiment several times. As shown in Figure 5.15, the increase in the packet size increases the EF average delay. We also note an increase in the EF jitter with an in crease of the BE packet size. As noted earlier, this variation is due to the presence of a BE packet already on service, when the EF packet arrives at the head of the queue. Figure 5.16 depicts the EF average one-way delay across LSP-2 at different volumes of background best effort cross traffic utilization (BE packet size =MTU).

Another crucial parameter in a P-FIFO based Diffserv architecture is the size of the buffer allocated to the EF queue. Different sizes of EF and BE packets may aggravate the situation. Accumulation may also get worse with multiple hops, as in the case of our test bed configuration. Hence, EF buffers need to have an appropriate size to accommodate the variability and avoid packet loss and excessive delays. In order to see how the EF buffer size affects the EF QoS performance, we ran the 5-minute MPEG-2 video clip a number of times while varying the buffer size (1 to 300 packets) in the presence of background best-effort traffic (maximum BE packet size/MTU was assigned). From Figure 5.17, it is evident that at the buffer size increases, the loss rate decreases. At a queue size of 3 packets close to 5% of the transmitted packets along the LSP were lost. For queue size of 30 packets only 0.1% of the transmitted packets didn't make it across, while no losses were noticed for queue size of 300.

To test the effect the number of hops along an LSP on the performance, we conducted tests on an LSP with 1, 2, and 3 hops (for the case of a single hop, we measure the LSR's forwarding delay and jitter). In all cases, the routers along the LSP are configured with identical settings. The EF delay and jitter are then measured at the ingress, and egress point of the LSP. As the graphs show in Figure 5.18, the average EF delay increases with the number of hops. This means that each hop adds its share of delay to the total EF delay. We also note an increase in the packet jitter with increasing number of hops.

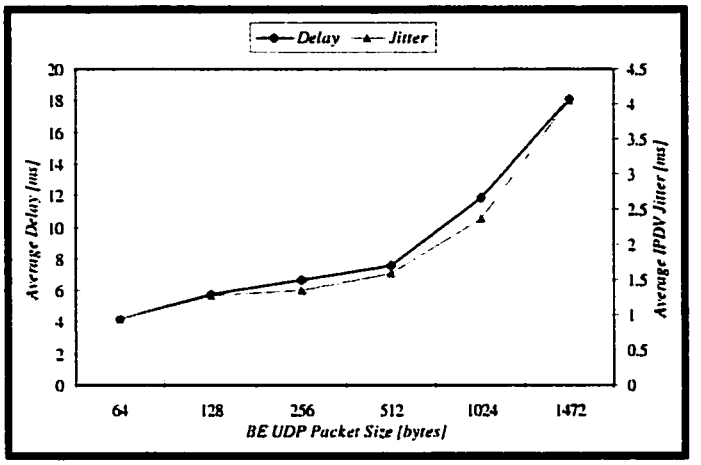

Figure 5.15 Video (EF) Average packet one-way delay and IPDV across LSP-2 versus BE packet size. (100% background traffic utilization, and Priority-FIFO- EF buffer size = 300 packets)

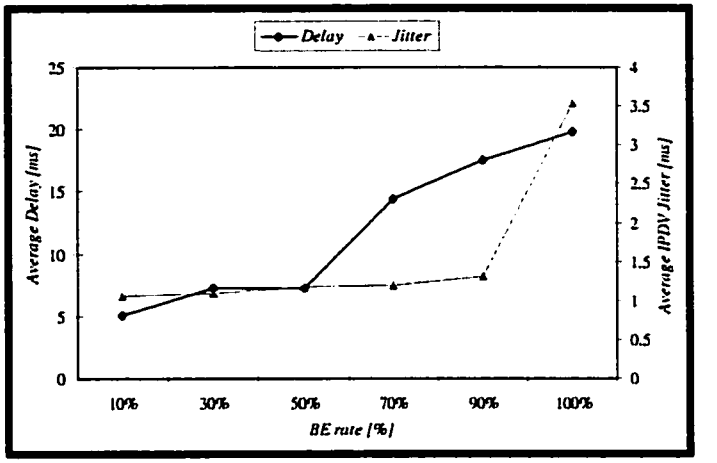

Figure 5.16 Video (EF) Average packet delay and IPDV across LSP-2 versus BE background traffic utilization. (BE packet size = 1472 bytes/MTU, and Priority-FIFO- EF buffer size = 300 packets)

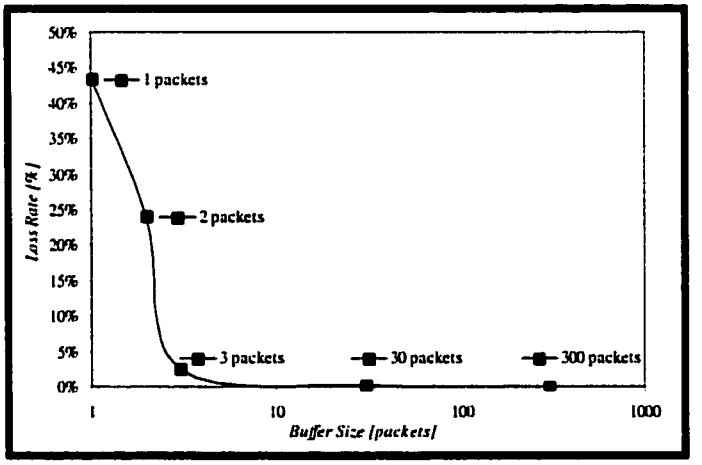

Figure 5.17 Packet loss rate of EF traffic across LSP-2 versus per node buffer size for Priority-FIFO.

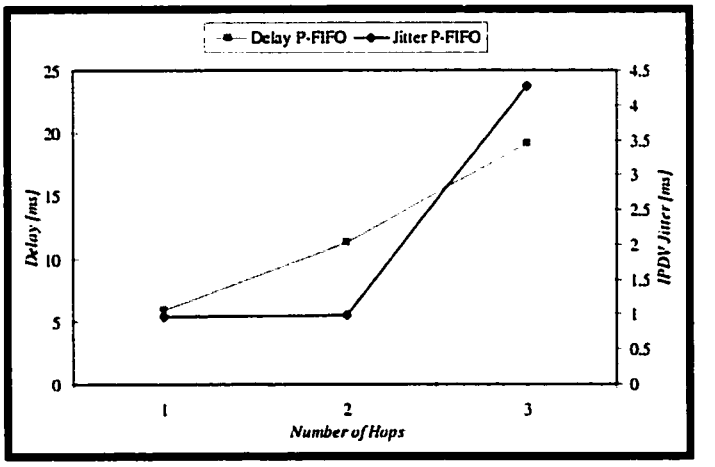

Figure 5.18 Average one-way delay and IPDV of EF traffic versus number of hops. (100% background traffic utilization, P-FIFO)

5.2.3 CBQ Based Differentiated Services:

We use a modified version of "tc" utility and CBQ scheduler running Linux kernel 2.4.17 that implements MPLS-DS capabilities. As mentioned earlier, CBQ allows for a variety of policies to be implemented among sibling classes (e.g. isolated, bounded, weight etc.). In our analysis, we present a typical CBQ configuration with two classes of traffic: a premium service (EF), and low priority (BE) service. The results presented are for the following Traffic Trunk (TT): configurations along the

established LSP-2: EF-TT with service rate reservation of 3 Mbps, and BE-TT of service rate 7 Mbps. The two Diffserv TTs are configured as isolated (i.e. they don't lend available bandwidth to each other), and bounded (i.e. they can't borrow available bandwidth from each other).

As shown in Figure 5.20, the increase in the BE background traffic load (10%-100% of BE-TT capacity) had a slight increase effect on the video stream's one-way average packet delay across LSP-2 (4 ms to 18 ms). The slight increase in this delay with the increase in the BE-traffic rate is due to the time spent to service at least one BE packet in the low-priority queue before going back to the high-priority queue. The CBQ scheduler (operates in Weighted Round Robin (WRR) among classes) and typically doesn't send any packet from the lower priority classes while there are backlogged packets in the higher priority queue (up to the max reserved bandwidth). This provides relative degree of protection for the delay-sensitive traffic (e.g. video applications, VoIP, etc.) when assigned a high priority class from non-delay sensitive competing traffic (e.g. WWW data traffic), typically assigned to a low priority. Figure 5.19 shows the effect of increasing the BE packet size on the EF average one-way delay, and IPDV across the LSP.

We also carried out tests similar to the case of P-FIFO, in order to check the effect of the number of hops on the total average delay and IPDV. Figure 5.21 shows that delays accumulate at each node across the LSP. We carried the measurements for different reservation rates for the EF class. We note that as the EF rate approaches the reserved rate at the Diffserv node (i.e. the EF class approaches the overlimit bound), EF packets start experiencing higher delays. This is because CBQ limits the class's bandwidth to the reserved rate (bounded class), and acts as a policer when a class goes over the reserved limit. On the other hand, in the case of P-FIFO, strict priority is given to the high priority class at all time with no regard to rate reservation, potentially, causing lower classes to starve out of their share of the link's bandwidth.

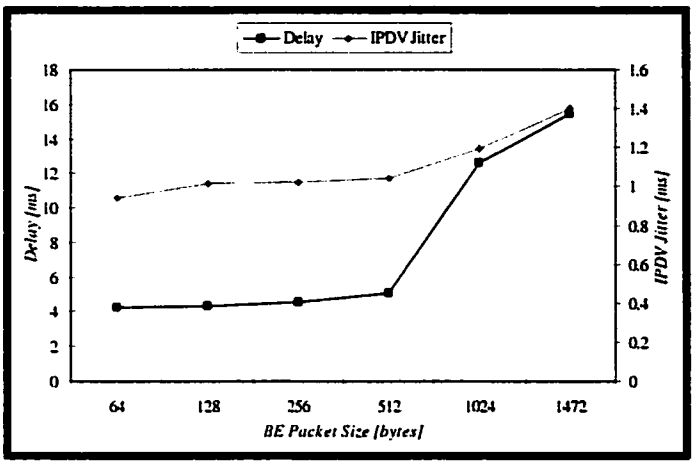

Figure 5.19 Average one-way delay and IPDV experienced by EF packets traveling across LSP-2 versus BE packet size. (100% background traffic utilization, and CBQ).

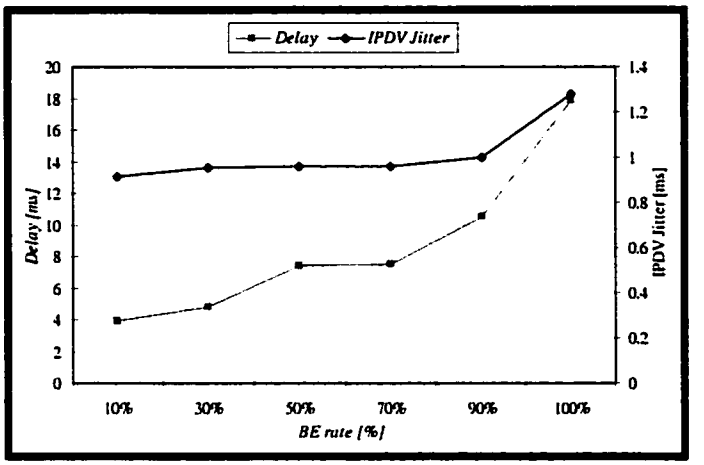

Figure 5.20 Average one-way delay and IPDV experienced by packets traveling along LSP-2 versus BE background traffic volume. (BE packet size = 1472 bytes/MTU, and Priority-FIFO).

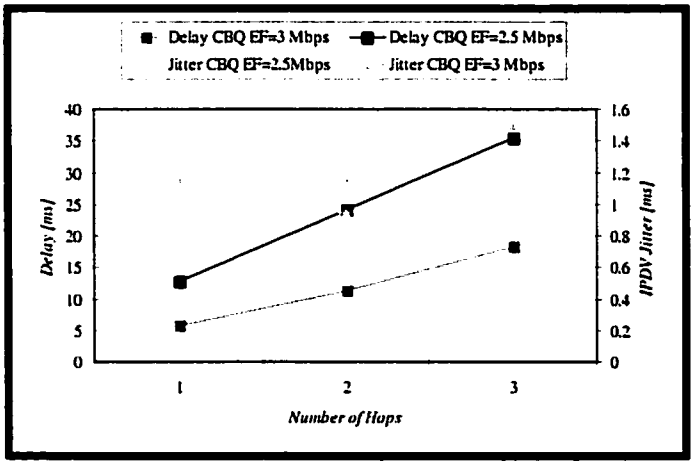

Figure 5.21 Average one-way delay and IPDV experienced by EF packets versus number of hops. (100% background traffic utilization, CBQ=2.5Mbps, 3Mbps)

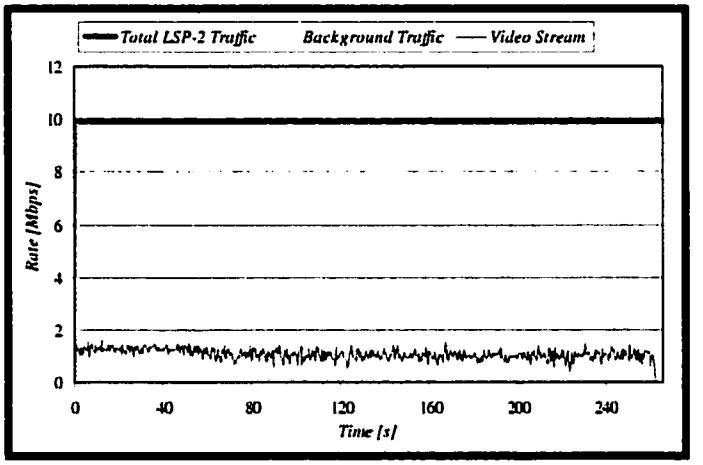

Figure 5.22 Video Stream Throughput (MPEG-2 clip) with no Diffserv on LSP-2.

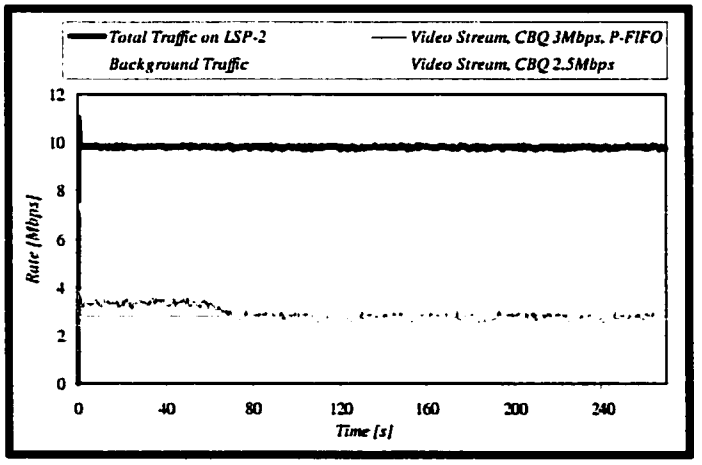

Figure 5.23 Throughput of the reference video stream (MPEG-2 clip) with Diffserv on LSP-2.

5.3 ALTERNATE LSP DYNAMIC ROUTING: GOLD LSP-2 (DIFFSERV-ENABLED) AND BRONZE LSP-1 (BEST-EFFORT LSP)

In the previous two sections, we studied the performance of a video stream when carried over a best-effort LSP, and a DS-enabled LSP. As it was expected and is known from earlier research, our results demonstrate that the best-effort LSP is unsuitable at times for the transmission of delay-sensitive traffic, especially in the presence of heavy background traffic utilization. On the other hand, the DS-enabled LSP offers preferential treatment to the high priority traffic when mixed with lower priority traffic, effectively isolating non-delay sensitive from delay sensitive traffic. However, we anticipate that such a service would be associated with a relatively higher service cost.

Sometimes it is desirable to use different LSPs for different classes of traffic. Hence, the physical network would be divided into multiple virtual networks, one per class. These virtual networks may have different topologies and resources. The end effect is that premium traffic can use more resources than best effort traffic. As well, premium traffic will have higher priority in getting the backup resources in the case of link congestion.

As mentioned earlier, MPLS-enabled networks facilitate the task of traffic engineering, and allow the configuration of explicitly signaled LSPs (ER-LSPs) that can be setup for different classes of service. Subsequently, it alleviates the problem of having all the traffic being routed on the shortest path, resulting in the over-utilization on some paths and underutilization on others. Moreover, Internet traffic exhibits a dynamic behavior over time, even when the LSPs are established with the bandwidth allocation made according to the traffic requirements existing at the LSP's establishment time. Hence, a mechanism to examine the degree of congestion-- especially on the best effort LSP where over-utilization might occur frequently-- and rapid redistribution of delay-sensitive traffic to a higher class LSP is essential.

In Chapter 3, we presented an overview on two approaches to carry out performance measurements in communication networks, namely the active, and the passive approach. The two

approaches have been adopted in several works such as [12], [72], [73], [74], and [76] to monitor the degree of utilization across certain Internet paths.

We establish the two LSPs (LSP-1, and LSP-2) connecting the MPLS network's endpoints with same configuration as before (refer to Figure 4.5). The DS-enabled "LSP-2" is configured as gold service, and is setup with two traffic trunks: a BE with an allocated bandwidth of 6Mbps, and an EF TT with allocated bandwidth of 4Mbps. LSP-1, on the other hand, is configured as bronze service with a total bandwidth of 10Mbps, and with no support of Diffserv.

The incoming traffic at the ingress point from remote network (172.16.10.0) destined to network (172.16.40.0) is divided into two categories: a normal priority cross-traffic that is always forwarded on LSP-1, and a premium high-priority traffic that is partitioned on LSP-1, and LSP-2.

5.3.1 Alternate LSP Routing using Active Probing

We saw that delay-estimate is a reliable metric, providing valuable information on the end-to-end performance and link congestion/failure state. We use an active probing approach similar to the one presented in [12] to monitor the delay on the best-effort LSP-1 as it becomes over-utilized, and switch the delay-sensitive traffic to LSP-2, which provides better performance guarantees at, possibly, a higher service cost.

The one-way delay on LSP-1 is obtained by transmitting probe packets at a rate of 2 packets/second between the ingress and egress endpoints. The probe packet (UDP) is time stamped at the ingress node at time T1, and arrives at the egress node at time T2. The packet is transmitted back to the ingress node stamped with the delay (T2-T1). To accommodate for lost probe packets, a maximum value of the delay is assumed after a timeout (500ms) if the reply datagram is not received. The choice of the probe packet size affects the serialization delay as discussed in Section 2.2, Table 2-2. The main source of delays in our network is due to the queuing of packets traversing different hops on the LSP. Figure 5.24 shows a slight increase in the OWD across LSP-1 (from 500μs-1.6ms) when increasing the size of the UDP probe packet (from 64 bytes to1472 bytes). No background traffic has been applied, in order to eliminate queuing delays generated by its presence. In our tests, we set the probe packet size close to the video packets' size (1000 bytes) in order to provide a closer indicator of the delay experienced by the video stream.

An estimate of the projected future delay is formed after each reading by using the estimator mentioned in Chapter 4 and discussed in detail in [1] [2]. The estimator's parameters were fine-tuned by carrying out several pilot runs on OPNET that are available in Appendix B. Figure 5.25 shows the probe packet delay, and the actual one-way delay experienced by packets crossing LSP-1 when 70% utilized with Poisson traffic. Figure 5.26 shows the output of the estimator that is generated after each probe delay reading.

We conducted experiments transmitting the MPEG-2 video flow (mean bit rate 2.8 Mbps) from Host A to Host B across the MPLS network. We injected aggregate Poisson traffic (mean bit rate 7Mbps) as background "cross-traffic" across LSP-1. As shown in Figure 5.27, the video stream's OWD delay increases up to the maximum allowable threshold (50ms in this case) when video packets are redirected towards LSP-2, which offers lower OWD.

Figure 5.28 shows the traffic breakdown into delay-sensitive non delay-sensitive, and total traffic across LSP-1 during the period of transmission of the video stream. Figure 5.29 shows the redirected fraction of delay-sensitive traffic onto LSP-2. At the egress router, video packets arriving from the two LSPs are forwarded to Host B, providing a continuous stream of packets at an acceptable throughput as shown in Figure 5.30. Figures 5.31 and 5.32 show the OWD and IPDV CDF of the video stream in the presence of 70% utilization of Poisson traffic on LSP-1: 1) when transmitted over on the best effort LSP and not applying alternate LSP routing, 2) when transmitted over the Diffserv-enabled LSP and not applying alternate LSP routing, and 3) when applying alternate LSP routing between LSP-1 and LSP-2. As shown, the Diffserv-enabled LSP offers the best performance in terms of the OWD and IPDV of the video stream. In the case of the best-effort LSP, as illustrated in Section 5.1, the OWD of the video packets could well rise over the 100 ms bound as the LSP becomes overutilized. When applying the alternate LSP dynamic routing algorithm with a delay threshold on the delay-sensitive traffic of 50 ms, we notice almost 85% of the video stream packets experience delays below the 50ms bound. The number of packets violating the delay bound can be further reduced by modifying the estimator to switch more portions the delay-sensitive traffic to LSP-2 as shown in Figure 5.33.

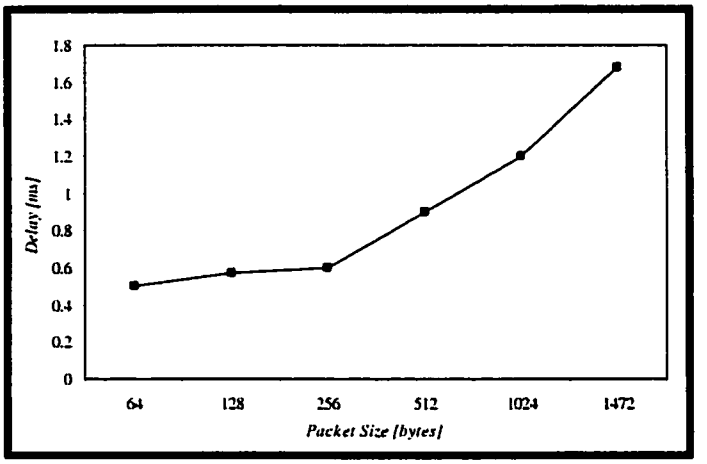

Figure 5.24 One-way probe packet delay versus probe packet size. (LSP-1, no background traffic)

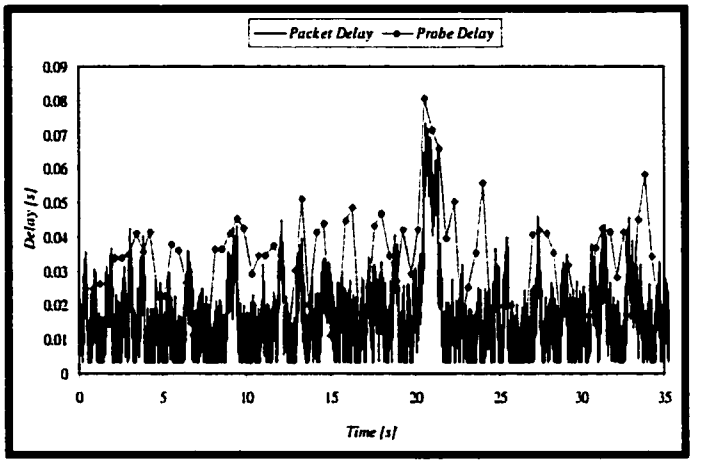

Figure 5.25 One-way packet delay and probe paket delay. (70% Poisson traffic on LSP-1)

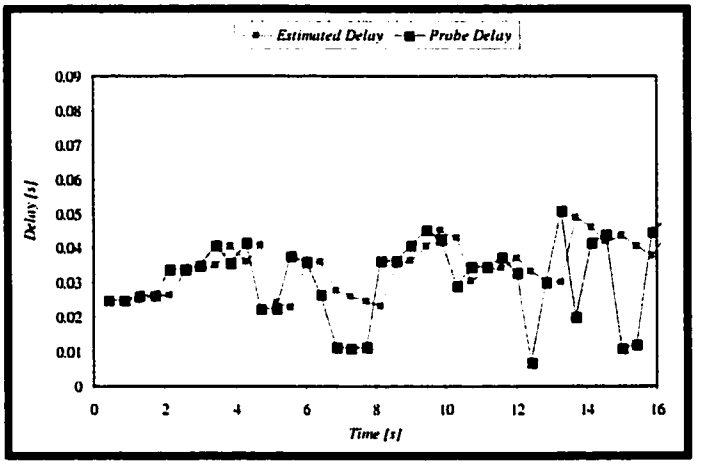

Figure 5.26 Estimated delay generated from probe packet delay. (70% Poisson traffic on LSP-1)

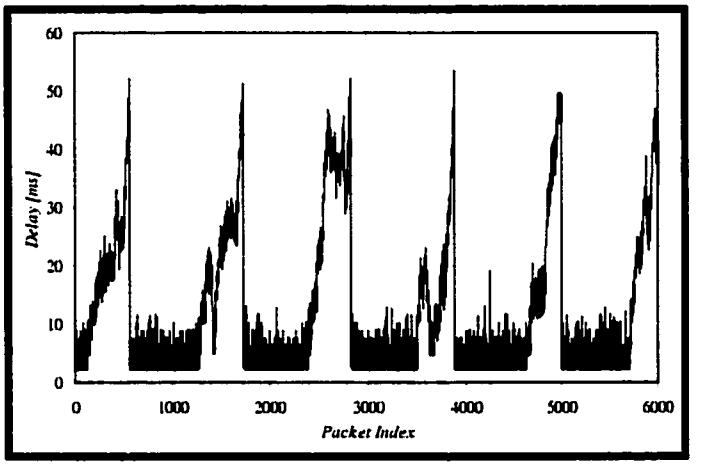

Figure 5.27 One-way delay of video stream with dynamic partitioning (70% Poisson cross-traffic utilization on LSP-1)

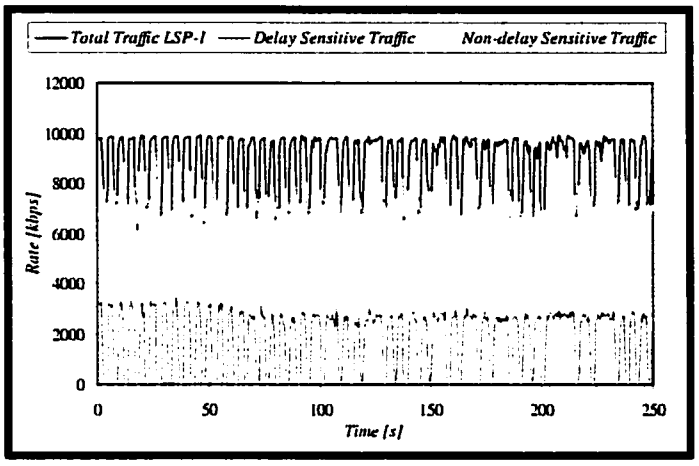

Figure 5.28 Traffic breakdown on LSP-1 with dynamic partitioning of delay-sensitive traffic (active monitoring).

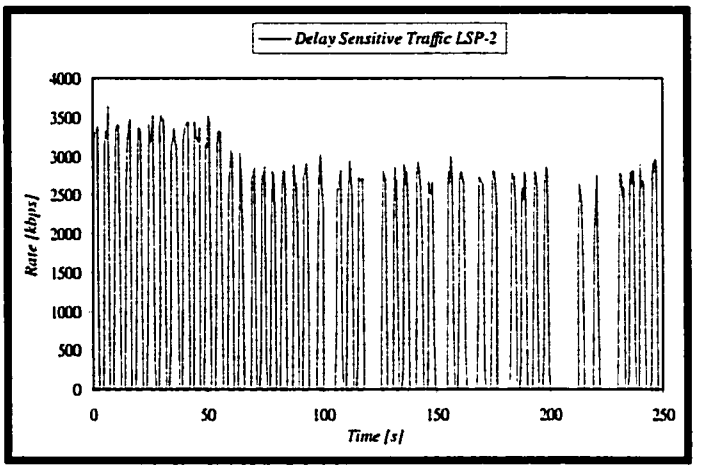

Figure 5.29 Fraction of video traffic forwarded on LSP-2.

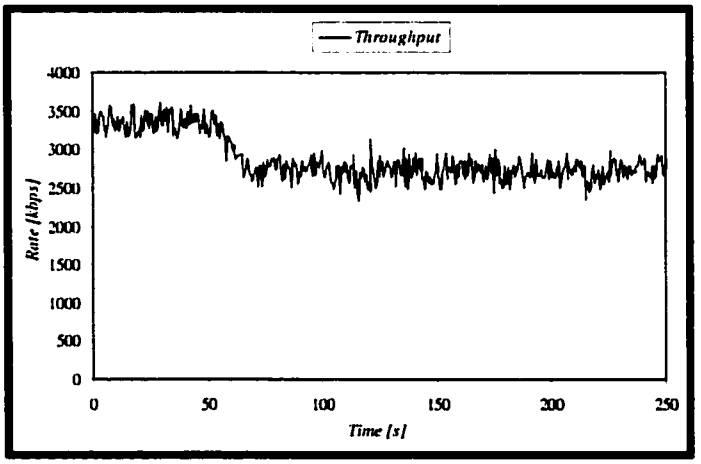

Figure 5.30 Video stream throughput at client Host B with dynamic partitioning of traffic (active probing).

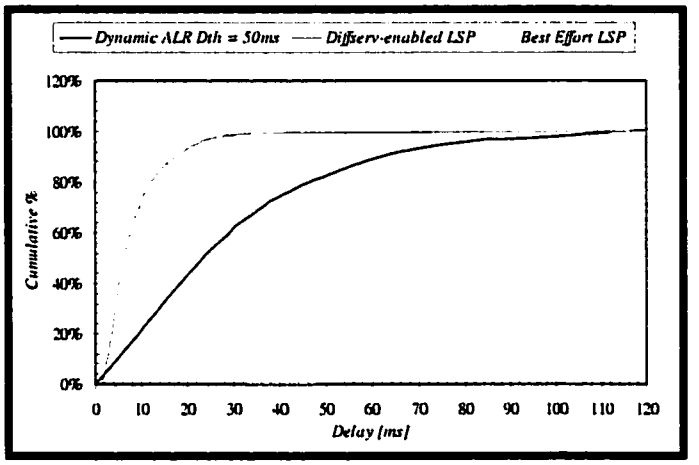

Figure 5.31 Percentage of reference stream's packets experiencing one-way packet delay under a certain value with 70 % utilization of Poisson traffic cross-traffic on LSP-1: 1) no ALR best-effort case, 2) no ALR diffserv, and 3) with dynamic ALR.

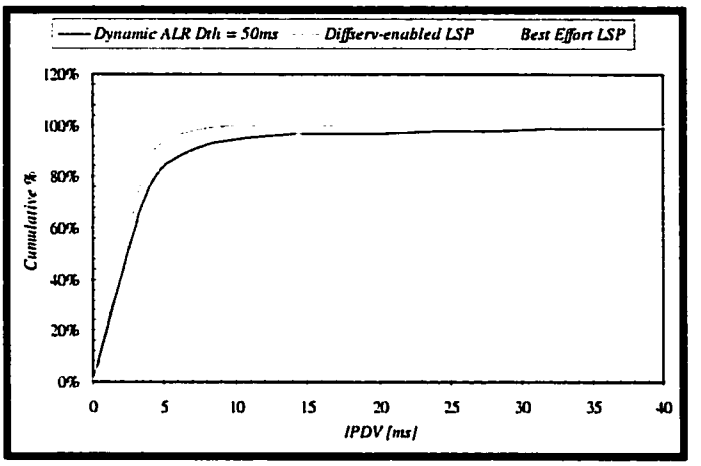

Figure 5.32 Percentage of reference stream's packets experiencing IPDV under a certain value with 70 % utilization of Poisson traffic cross-traffic on LSP-1: 1) no ALR best-effort case, 2) no ALR diffserv, and 3) with dynamic ALR.

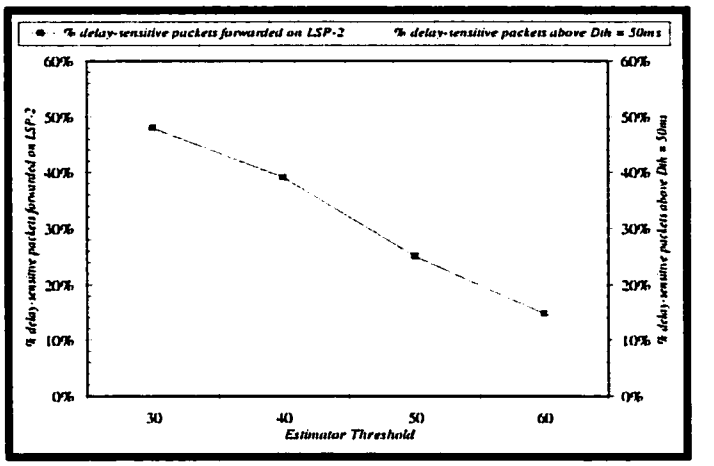

Figure 5.33 (1) Percentage of delay-sensitive traffic forwarded on gold LSP-2. (2) Percentage of delay-sensitive above delay bound = 50 ms. (70% Poisson cross-traffic on LSP-1).

5.3.2 Alternate LSP Routing using Passive Monitoring

The passive approach uses devices or mechanisms to watch the network traffic as it passes by. The devices can be special purpose devices such as a sniffer, or they can be built-in applications in routers, switches or end node hosts (e.g. Remote Monitoring (RMON) [65], Simple Network Monitoring Protocol (SNMP) [72], etc.).

The passive approach measures real traffic. It does not introduce traffic on the network though the transmission of probes, however, the polling, required to collect the passive data (e.g. counters, traps, alarms, etc.), generates network traffic that can be substantial.

We employed a similar technique on our testbed to monitor the incoming traffic at the ingress router, and infer the degree of congestion on the subject LSP. In other words, the ingress LER continuously monitors the total incoming traffic rate destined to an LSP. Once it is determined that the incoming traffic volume exceeds a threshold of the LSP capacity, excess traffic can be rapidly shifted to other preferred LSPs—possibly higher-grade classes. Traffic monitoring in Linux is possible by attaching an ingress firewall filter (fw) and a "policer" with a specific police-rate limit at the input device. The firewall policer tags excess packets with a mark that can later be used in making a routing decision to different LSPs.

In Section 5.1 we have shown that the one-way delay across the LSP increases as the LSP traffic utilization load increases. Hence, in order to ensure a low end-to-end delay, the traffic utilization across the LSP has to be kept below its maximum reserved bandwidth to ensure no congestion occurs at subsequent hops. Figure 5.34 shows the breakdown of the traffic flowing on LSP-1 (for policing rates of 10Mbps, and 9Mbps) when transmitting the video stream across the MPLS network with 7Mbps Poisson cross-traffic on LSP-1. As shown in Figure 5.33, the forwarded portion of the delay-sensitive traffic across LSP-1 (bronze service) decreases as the policing rate decreases, while the remaining part is forwarded across LSP-2 (gold service) as shown in Figure 5.35. Figure 5.37 shows the one-way delay experienced by the video packets of the reference stream when applying alternate LSP routing using the passive approach with policing rate of 10Mbps, and 9Mbps.

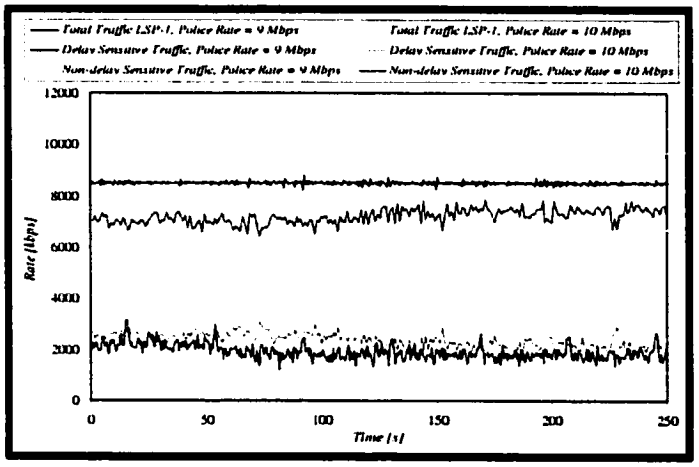

Figure 5.34 Traffic breakdown on LSP-1 with dynamic partitioning of delay-sensitive traffic (passive monitoring).

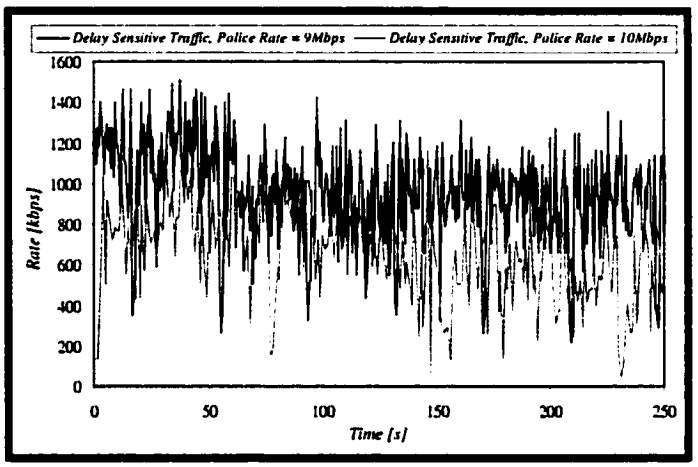

Figure 5.35 Fraction of video traffic forwarded on LSP-2.

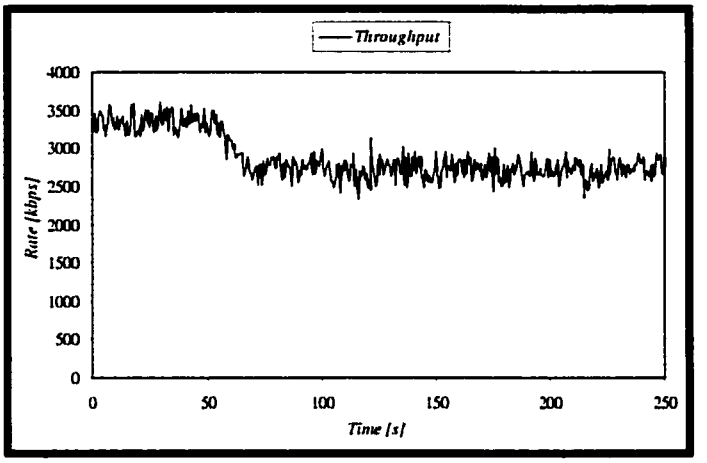

Figure 5.36 Video stream throughput at client Host B with dynamic partitioning of traffic (passive policing).

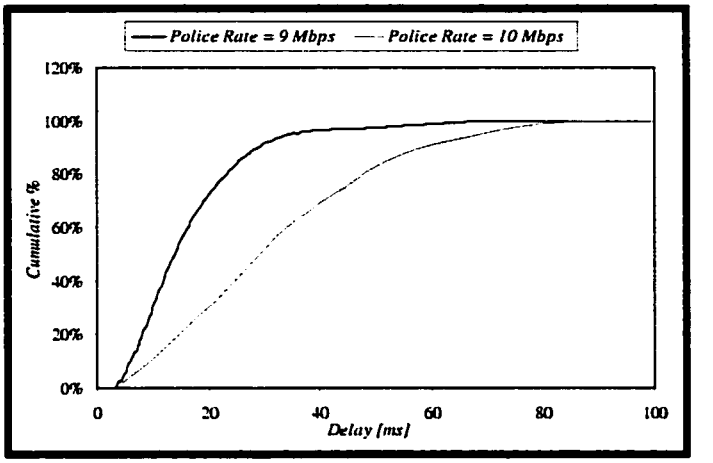

Figure 5.37 One-way delay of video stream (Police rate at ingress = 90, 100 % LSP capacity).

5.4 CONCLUSION

In this chapter we have carried out performance analysis of a video stream carried along LSPs with different QoS configurations. In section 5.1, results presented showed that on the best-effort LSP QoS parameters (delay, jitter, and loss) are affected by the degree of utilization across the LSP as well as the nature of traffic sharing the resources on the LSP (e.g. CBR, VBR, WWW data traffic, etc.).

In Section 5.2, we incorporated the Diffserv model on our MPLS testbed using two types of link sharing schedulers supported by Linux: 1) Priority-FIFO, and 2) CBQ. It is evident using both schedulers, that when the Diffserv EF classification is used for video, the effect of the background traffic diminishes and the video stream passes through without major complications. The improvements offered by Diffserv over best effort model become more evident, as the network load increases. In this case, the results obtained when integrating Diffserv with MPLS are in agreement with those found in the literature like [97], [98], etc. For example, with the P-FIFO scheme special admission control policy need be active at the ingress point so to ensure no starving of service is suffered by low priority traffic when mixed with the high priority type. Second, in the case where resource reservation is performed at each Diffserv node (e.g. CBQ scheduling), a dynamic bandwidth management scheme (e.g. Bandwidth Broker [65], or Cisco's automatic bandwidth adjustment [4]) that is capable of adapting the capacity of the TTs based on the offered load at the

ingress routers would be useful, in order to ensure guaranteed levels of QoS. While this could be thought of as a solution that can be applied with relatively long bandwidth re-adjustment periods, Alternative LSP Routing (ALR) can offers a effective solution for reacting to short-term periods of traffic fluctuations. As mentioned earlier, ALR protects time critical traffic by dynamically forwarding its packets to less loaded or Under-utilized LSPs (U-LSPs).

In Section 5.3, we have implemented the Cheapest Path First algorithm to partition the delay-sensitive traffic among multiple paths of different CoS. We used two approaches to monitor the performance on the best-effort LSP: 1) active probing approach, and 2) passive monitoring approach. The active approach resembled a closed loop feedback mechanism on the estimated delay experienced by traffic flowing through the LSP. On the other hand, the passive approach resembled an open-loop mechanism, whereby traffic is policed at the ingress point of the network before being forwarded on the LSP without feedback on the delay experienced thereafter.

Chapter 6

CONCLUSION AND FUTURE WORK

It is evident that providing strict deterministic service guarantees to delay-sensitive applications on the non-deterministic IP network is becoming a more challenging task as networks continue grow and gain new functionalities. The best-effort Internet model treated all users in the same way. In practice, however, users have different needs, use the network for different purposes, and are ready to pay for the service in different ways.

This Thesis analyzed the main blocks that, together, provide different levels of quality to different group of entities. The first part presented the theoretical foundations of the network Quality of Service, and the major components that have to be studied in order to provide guarantees to the end-users. It also provided an introduction to the different network models-- including Intserv, Diffserv, and MPLS -- that attempted to address the issue of providing network QoS guarantees to delay-sensitive applications.

MPLS provides the means to differentiate paths of different types of traffic based on a user defined filtering criteria at the edge of the network. Diffserv is able to differentiate between types of traffic on the same path based on the IP-defined DSCP field. The Differentiated services model, when combined with MPLS's traffic engineering capabilities, can provide a suitable architecture for ensuring QoS guarantees to various kinds of applications, especially when multiple parallel paths to the same destination exist in a network. This research showed how MPLS allows for the configuration of explicitly signaled LSPs (tunnels) with different configurations that can be established for different classes of service, and provided a dynamic scheme for rerouting delay-sensitive traffic to alternate LSPs, which is based on the Cheapest Path First Algorithm.

The second part of the thesis dealt with the implementation of the proposal on a testbed of Linux-based routers connecting remote networks across an MPLS network. The routers were setup and configured to support Diffserv and MPLS features. Explicit-route LSPs with different configurations were established, using the RSVP-TE signaling protocol. The Diffserv capabilities were implemented on the LSRs using two Linux-supported packet-scheduling algorithms: Priority-FIFO, and CBQ. The MPLS network was used to transport a video stream (marked with EF DSCP) originating from a Multimedia Server to a remote client. Various traffic models including Poisson traffic, CBR traffic, VBR traffic, and WWW traffic were developed and integrated into our testbed to study their effect when mixed with the video stream on different LSP configurations.

We carried out a performance analysis of a video stream across the best-effort LSP in the presence of the different types of background cross-traffic types. The results have shown that the QoS parameters (delay, jitter, and loss) are affected by the degree of utilization across the LSP as well as the type of traffic carried (CBR, VBR, WWW data traffic, etc.). Experiments were also conducted by transmitting the video stream (marked with EF PHB) on a Diffserv-enabled LSP, and a comparison between two packet-scheduling algorithms was presented. In the last section, the partitioning scheme across multiple LSPs was implemented using two approaches: active probing, and passive policing for the purpose of monitoring the performance on the LSPs, and re-route delay sensitive traffic to higher class LSP whenever needed.

Future Work

The results obtained in our study are not able to cover all the unsolved issues in the QoS arena. However, they represent a starting point for the research activities that the author will embark on in the future.

The research shed light on the short-term solution to reroute delay-sensitive traffic to an existing alternate LSP, as the performance on the current LSP is no longer suitable. However, the testbed will be further exploited to study the possibility of forecasting traffic-load changes and dynamically resizing the capacity of the LSP tunnels as a long-term solution to accommodate the changes of the incoming traffic over the lifetime of the LSP. This can be accomplished by

introducing central management authority in the domain (e.g. a Bandwidth Broker), or through the use of bandwidth estimators (e.g. Cisco's auto bandwidth adjustment tool) at the ingress LSR of the network.

The second area that this research sheds light on is the possibility of providing end-to-end QoS among multiple MPLS domains and the way QoS characteristics are propagated across multiple domains. This is particularly interesting when integrating the current testbed with a wireless domain running Hierarchical-MPLS [96] and/or Micro-Mobility MPLS [99], or an optical network running GMPLS.

REFERENCES

[1] Tarek Saad, Tingzhou Yang, Dimitrios Makrakis, Voicu Groza, "DiffServ-enabled Adaptive Traffic Engineering over MPLS," Proceedings of ICII2001-Beijing IEEE, November 2001.

[2] Tarek Saad, Tingzhou Yang, H. Yu, Dimitrios Makrakis, Voicu Groza, Emil Petriu, "QoS Support for Voice, and Video Applications over DiffServ-Enabled MPLS Networks," 8th International Conference on Advances in Communications and Control, Greece July 2001.

[3] Tarek Saad, Tingzhou Yang, Dimitrios Makrakis, Voicu Groza, "Inter-Domain Adaptive Traffic Engineering for IP Differentiated Services MPLS-based Networks," CCECE'02 Proceedings, Winnipeg, May 2002.

[4] Rekhter, Y., Davie, B., Katz, D., Rosen, E. and G. Swallow, "Cisco Systems' Tag Switching Architecture – Overview," RFC 2105, February 1997.

[5] P. Sharma, D. Estrin, S. Floyd, and V. Jacobson. "Scalable Timers for Soft State Protocols," 1997.

[6] V. Jackobson, "Interpacket arrival variance and mean." Letter to the TCP-IP mailing list, June 15, 1987.

[7] P. Karn, and C. Partridge, "Improving Round-Trip Time Estimators in Reliable Transport Protocols," ACM Transaction on Computer Systems, November 1991.

[8] S. Bakiras, and V. O. K. Li. "Quality of Service Support in Differentiated Services Packet Networks," ICC2001.

[9] E. Horlait, and N. Rouhana. " Differentiated Services, and Integrated Services use of MPLS," ISCC 2000.

[10] X. Xiao, A. Hannan, and B. Bailey, "Traffic Engineering with MPLS in the Internet," IEEE Network, March/April 2000.

[11] D. Awduche, "MPLS and traffic engineering in IP networks," IEEE Communications Magazine, December 1999.

[12] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering," IEEE, INFOCOM, Alaska, April 2001.

[13] Van Jacobson and Michael J. Karels, "Congestion Avoidance and Control," Proceedings of the Sigcomm'88 Symposium in Stanford, CA, August 1988.

[14] S. Nelakuditi, Z. Zhang, and R. Tsang. "Adaptive Proportional Routing: A localized QoS Routing Approach," InfoCom2000.

[15] "Ipsilon's General Switch Management Protocol Specification Version 2.0," RFC 2297, ftp://ftp.isi.edu/in-notes/rfc2297.txt

[16] OPtimized Network Engineering Tool (OPNET Technologies): http://www.mil3.com

[17] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus, "Requirements for Traffic Engineering over MPLS," RFC 2702, Sept. 1999.

[18] D. Awduche, A. Hannan and X. Xiao, "Applicability Statement for Extensions to RSVP for LSP-LSPs," Internet Draft, Sept. 1999. Work in progress.

[19] E. Rosen, A. Viswanathan and R.Callon, "Multiprotocol Label Switching Architecture," Internet Draft, Aug. 1999. Work in progress.

[20] L. Andersson, P. Doolan, N. Feldman, A. Fredette and B. Thomas, "LDP Specification," RFC 3036, January 2001.

[21] E. Crawley, R. Nair, B. Jajagopalan and H. Sandick, "A Framework for QoS-based Routing in the Internet," RFC 2386, Aug. 1998

[22] T. Li, G. Swallow and D. Awduche, "IGP requirements for Traffic Engineering with MPLS," Internet Draft, Feb. 1999. Work in progress.

[23] N. Shen and H. Smit, "Calculating IGP routes over Traffic Engineering LSPs," Internet Draft, June 1999. Work in progress.

[24] E. Rosen and Y. Rekhter, "BGP/MPLS VPN," RFC 2547, March 1999.

[25] K. Nichols, S. Blake, F. Baker and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474, Dec. 1998.

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

[27] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889: "RTP: A Transport Protocol for real-time applications," January 1996.

[28] "Measurements and Analysis of End-to-End Internet Dynamics," Vern Paxson. Ph.D. Thesis, Computer Science Division, University of California, Berkeley

[29] Francois Le Faucher, Bruce Davie, et al. "MPLS support of Differentiated Services," Internet Draft March 2000. Work in progress.

[30] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services," Internet Society, RFC 2475.

[31] R. Woundy, A. Viswanathan, N. Feldman, and Rick Boivie, "ARIS: Aggregate Route-based IP Switching," IETF Internet Draft, November 1996. Work in progress.

[32] J. Postel, "Daytime Protocol," RFC 867, May 1983.

[33] J. Postel, and K. Harrenstien, "Time Protocol," RFC 868, May 1983.

[34] David L. Mills, "Network Time Protocol (Version 3): Specification, Implementation and Analysis," March 1992.

[35] D. Awduche, L. Berger, D-H. Gan, T. Li, G. Swallow, and V. Srinivasan, "RSVP-TE Extensions to RSVP for LSP Tunnels," Internet Draft, 2000. Work in progress.

[36] The Internet Engineering Task Force (IETF), http://www.ietf.org/

[37] R. Braden, L. Zhang, S.Herzog, and S. Jamin, "Resource ReServation Protocol (RSVP) – Version 1Functional Specification," IETF RFC2205, September 1997.

[38] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, E. Felstaine, "A framework for end-to-end QoS combining RSVP-Intserv and Differentiated Services," IETF Internet Draft, May 2000. Work in progress.

[39] IETF "Multiprotocol Label Switching Working Group." http://www.ietf.org/html.charters/mpls-charter.html

[40] IETF, "Integrated Services (Intserv) Working Group," http://www.ietf.org/

[41] IETF "Differentiated Services (Diffserv) Working Group." http://www.ietf.org/html.charters/diffserv-charter.html

[42] IETF "Internet Traffic Engineering (tewg) Working Group," http://www.ietf.org/html.charters/tewg-charter.html

[43] IETF "Internet Protocol Performance Metrics (ippm) Working Group," http://www.ietf.org/html.charters/ippm-charter.html

[44] "Iperf: Internet Performance Measurement Tool," http://dast.nlanr.net/Projects/Iperf/

[45] "Active Measurement Program, NLNAR," http://amp.nlanr.net/AMP/

[46] "Surveyor", http://www.advanced.org/csg-ippm/

[47] "Ethereal Protocol Network Analyzer Tool," http://www.ethereal.com/

[48] S. Floyd, "Notes on CBQ and Guaranteed Service," available from http://www.aciri.org/floyd/papers/guaranteed.ps, July 1995.

[49] "Traffic Engineering for Quality of Service in the Internet at Large Scale (TEQUILA)," http://www.ist-tequila.org/

[50] "RIPE NCC's Test Traffic Measurements Service," http://www.ripe.net/test-traffic/

[51] "End-to-End Internet Delay Dynamics," IEICE Technical Report of CQ WG, May 2000.

[52] D. Goderis et al, "Service Level Specification Semantics and Parameters," draft-tequila-sls-00.txt, November 2000. Work in progress.

[53] B. Davie, J. Lawrence, K. McCloghrie, E. Rosen, G. Swallow, Y. Rekhter, and P. Doolan, "MPLS using LDP and ATM VC Switching," RFC 3035, January 2001. ftp://ftp.isi.edu/in-notes/rfc3035.txt

[54] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching," Architecture Network Working Group RFC 3031, January 2001.

[55] "IP Navigator MPLS," http://www.lucent.com

[56] J. Postel, "Internet control message protocol," RFC 792, IETF, September 1981

[57] "RSVP-TE daemon for DiffServ over MPLS under Linux," http://dsmpls.atlantis.rug.ac.be/

[58] "MPLS for Linux," http://sourceforge.net/projects/mpls-linux/

[59] Bernard Fortz, and Mikkel Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," IEEE INFOCOM(2) 2000.

[60] D.J. Goodman et. Al., "Efficiency of Packet Reservation Multiple Access," IEEE Transaction on Veh. Tech. Vol. 40, no. 1, pp. 170-176, February 1991.

[61] Sverre H. Huseby, "Video on the World Wide Web, Accessing Video from WWW Browsers," Technical Report, University of Oslo, February 1997.

[62] S. Shalunov, B. Teitelbaum, and M. Zekauskas, "A One-Way Delay Measurement Protocol." IPPM wok in progress, IETF 2001.

[63] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," IEEE RFC 2597, June 1999.

[64] C. Demichelis, and P. Chimento, "Instantaneous Packet Delay Variation Metric for IPPM," ippm draft. Work in progress.

[65] Panos Trimintzios et al., "A Management and Control Architecture for Providing IP Differentiated Services in MPLS-Based Networks," IEEE Communications Magazine, May 2001.

[66] A.J. McGregor, "The IP Measurement Protocol," Availableonline at http://moat.nlanr.net/AMP/AMP/IPMP/, 1998..

[67] J. Postel, "User Datagram Protocol," RFC 768, August 1980.

[68] J. Postel, "Transmission Control Protocol," RFC 793, September 1981.

[69] Tarek N. Saadawi, Mostafa Ammar, Ahmed El Hakeem, "Fundamentals of telecommunication networks," Wiley Series in Telecommunications and Signal Processing, September 1994.

[70] S. Vutukury and J.J. Garcia-Luna-Aceves, "A Traffic Engineering Approach based on Minimum-Delay Routing," Proc. IEEE IC3N 2000, Las Vegas, Nevada, USA, October 16--19, 2000.

[71] A. M. Zoubir and B.Boashash, "The Bootstrap and Its Application in Signal Processing", IEEE Signal Processing Magazine, January 1998.

[72] J. Cleary, S. Donnelly, I. Graham, A. McGregor, and M. Pearson, "Design principles for accurate passive measurement," in PAM 2000, The First Passive and Active Measurement Workshop, (Hamilton, New Zealand), April 2000.

[73] J. C. Bolot, "Characterizing end-to-end packet delay and loss in the internet," Journal of High-Speed Networks, vol. 2, pp. 305-323, September 1993.

[74] V. Paxson, "End-to-end internet packet dynamics," in Proceedings of ACM Sigcomm'97, (Cannes, France), pp. 139-152, September 14-18 1997.

[75] V. Jacobson, Pathchar - a tool to infer characteristics of Internet paths, available at: http://www.employees.org/bmah/software/pchar/ ed., 1997.

[76] L. Cottrell, "Comparison of some internet active end-to-end performance measurement projects," SLAC, http://www.slac.stanford.edu/comp/net/wan-mon/iepm-cf.html, July 1999.

[77] J. C. Bolot, "Characterizing End-to-End Packet Delay and Loss in the Internet," Journal of High-Speed Networks, vol. 2, pp. 305-323, September 1993.

[78] V. Paxson, "End-to-End Internet Packet Dynamics," in Proceedings of ACM Sigcomm'97, (Cannes, France), pp. 139-152, September 1997.

[79] W. Matthews and L. Cottrell, "The Pinger Project: Active Internet Performance Monitoring for the henp Community," IEEE Communications Magazine special issue on "Network Traffic Measurements and Experiments," May 2000.

[80] Esmael Dinan, Daniel Awduche, and Bijan Jabbari, "Analytical Framework for Dynamic Traffic Partitioning in MPLS Networks", ICC 2000.

[81] T. Berners-Lee, R. Fielding, and H. Nielsen. Hypertest Transfer Protocol- HTTP/1.0. RFC 1945, May 1996.

[82] Bruce Mah, "An Empirical Model of HTTP Network Traffic", INFOCOM, April 1997.

[83] Bruce Mah, Peter Sholander, Luis Martinez, and Lawrence Tolendino, " IPB: An Internet Protocol Benchmark Using Simulated Traffic." MASCOTS, July 1998.

[84] Peter Danzig, and Sugih Jamin, "tcplib: A Library of TCP Internetwork Traffic Characteristics." Technical Report USC-CS-91-495, Computer Science Department, University of Southern California, Los Angeles, CA, 1991.

[85] Vern Paxson, and Sally Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling", July 1995.

[86] Poul E. Heegaard, "GenSyn - a Java based generator of synthetic Internet traffic linking user behaviour models to real network protocols", ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, Sep 18-20, 2000, Monterey, CA (USA).

[87] "tcpdump and libcap ", http://www.tcpdump.org/

[88] "MPEG-4 and H.263 Video Traces for Network Performance Evaluation", http://www-tkn.ee.tu-berlin.de/~fitzek/TRACE/trace.html

[89] "MPEG Traces", http://viscom.kaist.ac.kr/video-trace.html

[90] The Multi-Generator (MGEN) Toolset, Naval Research Laboratory (NRL), http://manimac.itd.nrl.navy.mil/MGEN/

[91] "VideoLAN Open Source Video streaming", École Centrale Paris http://www.videolan.org/

[92] "Windows Media Technologies", http://www.microsoft.com/windows/windowsmedia/default.asp

[93] "bbMPEG", http://members.cox.net/beyeler/bbmpeg.html

[94] "Real Time Streaming Protocol," http://www.rtsp.org/

[95] "TE on Linux", Fujitsu Laboratories Ltd., http://www.labs.fujitsu.com/free/te-on-linux/

[96] T. Yang, and D. Makrakis, "Hierarchical Mobile MPLS: Supporting Delay Sensitive Applications over Wireless Internet", International Conference on Info-Tech & Info-Net (ICII 2001), Beijing, China, October 2001.

[97] T. Yang, Z. Chen, A. Hafid, and D. Makrakis, "An Evaluation of Different Marking Algorithms in Assured Forwarding Networks", IFIP/IEEE MMNS's 2000 conference, Fortaleza, Ceara, Brazil, 26 –28 September, 2000.

[98] T.Ferrari, G.Pau, C.Raffaelli , "Measurement Based Analysis of Delay in Priority Queuing," Proceeding of IEEE Global Telecommunication Conference Globecom 2001, S.Antonio, Texas; Nov 25-29 2001

[99] T. Yang, Y. Dong, Y. Zhang, and D. Makrakis, "Practical Approaches for Supporting Micro Mobility with MPLS", International Conference on Telecommunications 2002 (ICT 2002), Beijing, China, June 23, 2002.

[100] "Understanding Delay in Packet Voice Networks." CISCO white paper from WWW: http://www.cisco.com/warp/public/788/voip/delay-details.pdf

[101] M. Mathis, "Empirical Bulk Transfer Capacity," Internet Draft draft-ietf-bmwg-ippm-treno-btc-01.txt, July 1997

[102] Tingzhou Yang, Yixin Dong, Tarek Saad, Dimitrios Makrakis, "Dynamic Ubiquitous Services retrieval (DUST) System: Enabling Seamless Service Provision Support to Pervasive Users" Wireless and Optical Communications 2002, July 2002

APPENDICES

Appendix A

```
#!/bin/sh

################ Mark video traffic with DSCP 0x2e (EF PHB)  ################
./iptables -A PREROUTING -t mangle -p udp --dport 2002 -j FTOS --set-ftos  0x2e

################ Mark second flow with DSCP 0x2e (EF PHB)  ################
./iptables -A PREROUTING -t mangle -p udp --dport 2002 -j FTOS --set-ftos  0x2e

################ Mark all other flows with DSCP 0x00 (BE PHB)  ################
./iptables -A PREROUTING -t mangle -p udp --dport 2002 -j FTOS --set-ftos  0x00

################ Map DSCP 0x2e (EF PHB) on LSPID=12 with EXP=1 ##########
./tunnel -m -x 0x2 -d 172.16.40.2/32 -l12/1
```

Figure A. 1 Steps followed to mark incoming packets generated by the video server with the EF DSCP (0x2e) before pushing them into the MPLS tunnel. (iptables (version 1.2.4) was used for this purpose). Packets with DSCP 0x2e and destined to host B on LSPID= 12 are assigned an EXP value 1.

```
##### Set up proper qdisc on interface

$TC filter add dev $i parent 1:0 protocol mpls prio 1 tcindex mask 0xfc shift 2 pass_on
# Second a CBQ qdisc is used in order to support EF, and BE classes
$TC qdisc add dev $i parent 1:0 handle 2:0 cbq bandwidth 100Mbit cell 8 avpkt 1000 mpu 64
$TC filter add dev $i parent 2:0 protocol mpls prio 1 tcindex mask 0xf0 shift 4 pass_on

## Definition of the CBQ leaf classes to support EF and BE

## EF class setup
echo EF
$TC class add dev $i parent 2:0 classid 2:5 cbq bandwidth 100Mbit rate $EF_RATE avpkt 1000 prio
1 bounded allot 1514 weight 1Mbit maxburst 10 defmap 0
$TC qdisc add dev $i parent 2:5 pfifo limit $EF_BUFFERSIZE
$TC filter add dev $i parent 1:0 protocol mpls prio 1 handle 0x2e tcindex classid 1:151
$TC filter add dev $i parent 2:0 protocol mpls prio 1 handle 5 tcindex classid 2:5

## BE class specific setup
echo BE
$TC class add dev $i parent 2:0 classid 2:6 cbq bandwidth 100Mbit rate $BERATE avpkt 1000 prio
7 allot 1514 weight $BE_RATE maxburst 21
$TC qdisc add dev $i parent 2:6 pfifo limit $BE_BUFFERSIZE
$TC filter add dev $i parent 1:0 protocol mpls prio 1 handle 0x0 tcindex classid 1:161
$TC filter add dev $i parent 2:0 protocol mpls prio 1 handle 6 tcindex classid 2:6
```

Figure A. 2 Sample Linux "tc" script to implement Diffserv EF and BE PHBs.

```
#include "unp.h"
```

```
#include "../util/error.c"
#include "../util/wrapsock.c"


void dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen);

int main(int argc, char **argv)
{
        int sockfd;
        struct sockaddr_in servaddr;
        FILE *cf;



        if (argc!=3)
                err_quit("usage: udpcli <IPaddress><trace_file>");

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family= AF_INET;
        servaddr.sin_port = htons(5000);

        if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr)<=0)
                err_quit("inet_pton error for %s", argv[1]);

        if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0))<0)
                err_sys("socket error");

    if ((cf=fopen( argv[2], "r")) == NULL )
                err_sys("File could not be opened");

        dg_cli(cf, sockfd, (SA *) &servaddr, sizeof(servaddr));
        fclose(cf);
        exit(0);

}

void dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen)
{
        int m,rate, seg_size, slpt, slpf;
        int i,i1, j1, n1,  j, n;
        int  count, rem,count2, rem2,  rem_seg;
        char *cbr, *vbr, *cbr1, *vbr1;

        cbr= (char*)malloc(1024);
        for (i=0; i< 1024; i++)
                cbr[i] = '0';

        fscanf(fp, "%d", &rate);
    while ( !feof(fp)) {
                count = rate/1024;
                rem = rate%1024;
                slpf = rate*8/100;     //transmit time for this frame microsecond
                slpt = 40000 - slpf;   //left time before transmit second frame
                vbr= (char*)malloc(rem);
        for (j=0; j< rem; j++)
                 vbr[j] = '0';
        for (n=0; n<count; n++)
              sendto(sockfd, cbr, strlen(cbr), 0, pservaddr, servlen);

                sendto(sockfd, vbr, strlen(vbr), 0, pservaddr, servlen);
                usleep(slpt);
        free(vbr);
                fscanf(fp, "%d", &rate);
     } // end of file while loop

    free(cbr);
}
```

Figure A. 3 C source-code: mpeg trace-files synthetic traffic generator

```
import java.lang.*;
import distributions.*;
import java.io.*;
class Driver extends Thread {
        long sleepTime;
        public Driver () {
                super();
                Client cls=null;
                try {
                                new Client().start();
                        }
                catch (Exception e){
                                e.printStackTrace();
                }
                System.out.println("END");
        }
        public void run(){
                double i, k, exponential, time1, r=3;
                ExponentialDistribution exp = new ExponentialDistribution(r);
                RandomVariable arrivalTime1 = new RandomVariable(exp, "X");
                int j=0;
                while (true)
                {
                                time1 = - Math.log(1 - Math.random()) / r; /* Exponential */
                                arrivalTime1.setValue(time1);
                                exponential = arrivalTime1.getValue();
                                sleepTime = ((long)(exponential*1000));
                                j+=1;
                                System.out.println("   Exponential ["+j+"]:  "+ sleepTime);
                                try
                                {
                                        this.sleep(sleepTime);
                                        new  Client().start();
                                }
                                catch (Exception exc){}
                }
        }

        public static void main(String[] args) {
                Driver t1 = new Driver();
                t1.start();
        }
}
```

Figure A. 4 Java source-code: WWW client sessions generator (exponential distribution)

Appendix B

OPNET Simulations

OPNET [16] is a simulation tool that provides a comprehensive development environment supporting the modeling of communication networks and distributed systems. OPNET allows large numbers of closely spaced events in sizeable network to be represented accurately. It uses a

modeling approach where networks are built of nodes interconnected by links. Each node's behavior is characterized by the constituent components. The components are modeled as a state-transition diagram.

We use OPNET as our simulation environment to validate our proposal, and achieve preliminary results on its applicability. We further implement a prototype network of Linux routers and carry out tests with real traffic sources to evaluate our model in a realistic environment.

Simulated Network Model

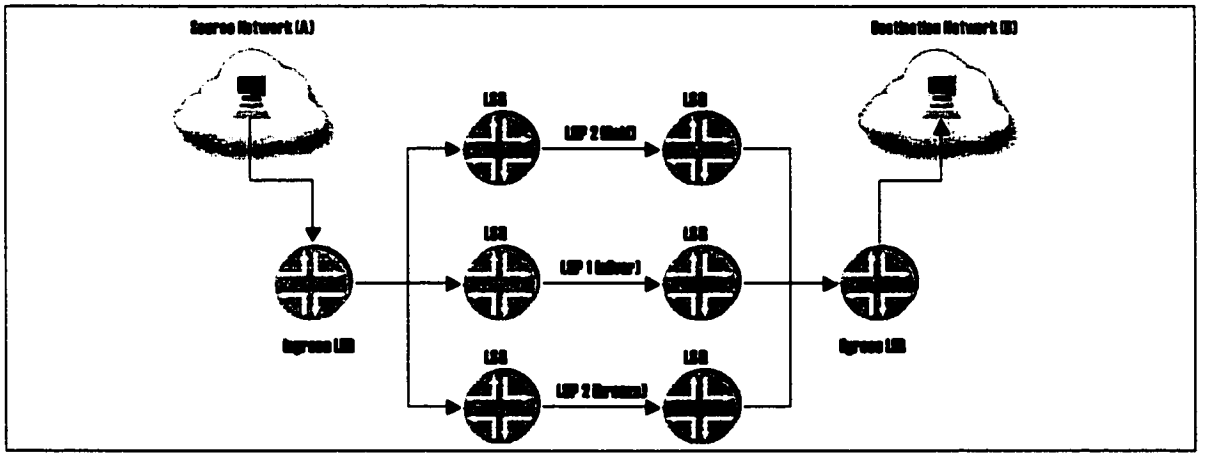

Figure B.1 Simulated network model architecture

We run a number of simulation tests on a network model consisting of two distant enterprise networks that are connected via an MPLS backbone network. The tests were carried out to validate and repeat some of the test results obtained from the testbed.

The MPLS network consists of LERs, and core LSRs. We assume that the source hosts will stamp the respective IP DSCP field that will be inspected at the edge router for service differentiation purposes. Three parallel LSPs are setup from the ingress LER to the egress LER, each of maximum capacity of 4 Mbps, and connecting the source and destination customer networks. We assume that the LSPs are initially setup such that their capacities can at least handle all traffic load crossing from the Ingress edge router and destined to the Egress edge router.

LSP_b (bronze) is configured as a best effort LSP that provides no special treatment for different classes of traffic, and can become at certain times over-utilized. LSP_s (silver) is configured

with two TTs: EF with capacity of 2Mbps, and a BE with capacity of 2Mbps. LSP_g (gold) is also configured with two TTs: EF with a capacity of 2 Mbps, and best effort with a capacity of 2Mbps. For our simulations, we are specifically interested in evaluating our CPF algorithm at achieving minimal service cost under the delay constraints set above, hence, we consider the case of infinite queues at the MPLS switch routers. We assume the cost service usage of the EF TT along the gold, silver, and bronze LSP are defined such that: $C_g > C_s > C_b$.

In our simulations we model the user's time-critical traffic with the Diffserv defined Expedited Forwarding (EF) Per Hop Behavior (PHB) [26]. The EF class was designed to carry traffic with low delay and virtually no loss without per-flow queuing. As discussed in Chapter 2, latency is mostly due to queuing experienced while traversing nodes in the network.

Simulation Results

We inject an aggregate of traffic classified as EF with a Poisson distributed rate of 1.5 Mbps, and an aggregate of traffic classified as BE with a Poisson distribute rate of 3 Mbps both running over UDP, originating from source network A and to destination network B, and traversing our MPLS backbone network. We specify two types of traffic in our simulator: engineered time-critical traffic, and standard best effort cross traffic. The engineered traffic is the time sensitive EF traffic that needs to be dynamically partitioned based on the CPF partitioning algorithm, and the cross traffic is the standard best effort background traffic and is assigned a fixed route through the bronze LSP.

We assume that the user specifies a certain upper bound D_{th} on the average end-to-end delay that the time-critical traffic can endure across the MPLS network. This value is used in our CPF algorithm to make a decision on rerouting the portion λ_p (refer to Section 3.3) of the time-critical traffic at each scheduling round after checking the estimated value of the delay on each LSP. We set the weight of λ_p to 5 packets. The value of λ_p as mentioned in [12] must be chosen of reasonable granularity to avoid packet reordering problems caused by rerouting consecutive packets of the same flow onto different LSPs.

Active test measurement traffic is performed by periodically sending a stream of probe packets that will discover the delay along each LSP in the MPLS network. The frequency of the test traffic

should not excessively overload the network, while still reflecting an accurate measure on the delay. As mentioned earlier, measurement projects typically use probe streams of average bandwidth lower than 10Kbps [76]. In our case, we assign a 100 ms time interval between two consecutive probes. We further employ the estimator algorithm, described in Section 3.5, to provide us with a quick estimate of the future time slot's delay that can be experienced on the EF-TTs on each of the three LSPs.

Figure B.2 shows customer (A)'s BE one-way delay across the MPLS network in Cumulative Distribution Function (CDF) plot. As mentioned, this cross traffic is constantly routed along LSPb with no consideration to the delays experienced. This translates to higher delays as LSP_b becomes overutilized—as shown, more than 60% of packets suffer delays higher than 400 ms. Figure B.3 shows customer (A)'s EF class traffic in CDF plot. The assumed upper bound on the delay across the MPLS network in this case is (D_{th}=120 ms). As shown, about 58% of the packets experienced delays below 120ms. In this case, packet delays higher than the D_{th} were experienced due to our TE-system being late to switch to the higher grade LSP.

We modified our estimator by assigning a higher value for psi (0.5) as the delays on the current LSP approach the user's specified Dth (Dsafe=0.8 Dth). This made our algorithm react quicker to switch to higher grade LSPs as probe packet delay approaches D_{th}. Figure B.5 shows the improvement achieved after making modification to the estimator. Figure B.6 shows the breakdown of customer (A) EF traffic distribution along the three LSPs, before and after applying the modification to the estimator. As seen, the total traffic is divided among the three LSPs. In the first case (before estimator modification), we notice that almost 30% of the total EF traffic went through on LSP_b, while in case 2 (after modifying the estimator), we see that only 15% went through on the same LSP_b. Of course, this meant better performance in terms of delays experienced per packet, but on the other hand, it also indicated a possibly higher service cost due to shifting higher portion of the traffic to LSP_s.

In Figure B.7, we show two cases for the customer assigning D_{th}=120ms, and 60 ms after modifying our estimator. Certainly, with lower D_{th} less traffic will be forwarded along LSP_b, while most of it will be shifted to the other two LSPs.

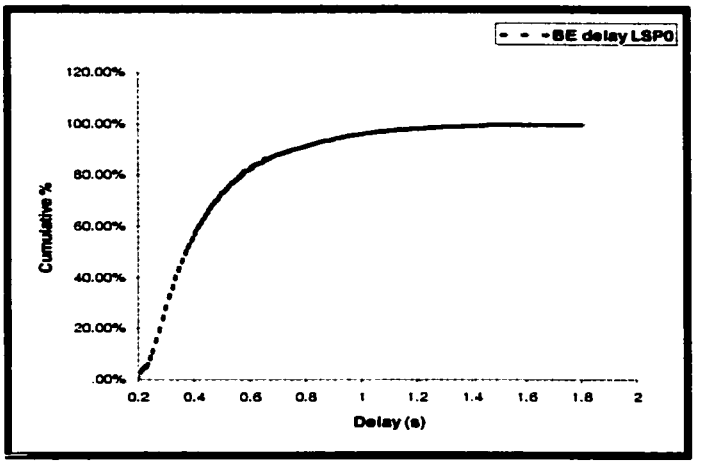

Figure B. 2 Customer (A) CDF for BE packet one-way on LSPb

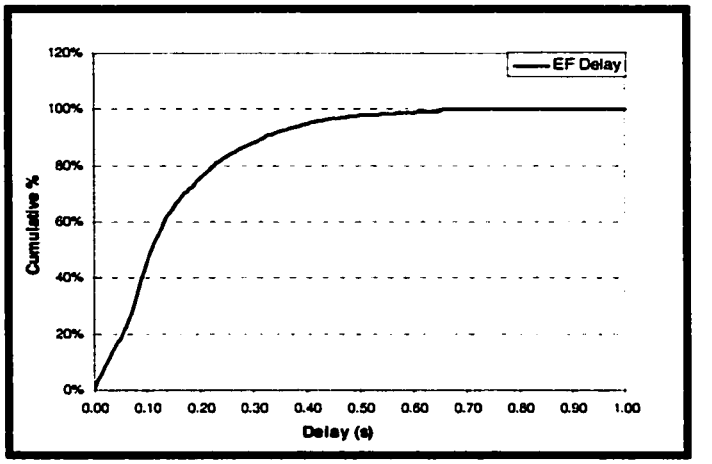

Figure B. 3 Customer (A) CDF for EF packet one-way delay for Dth= 120 ms.

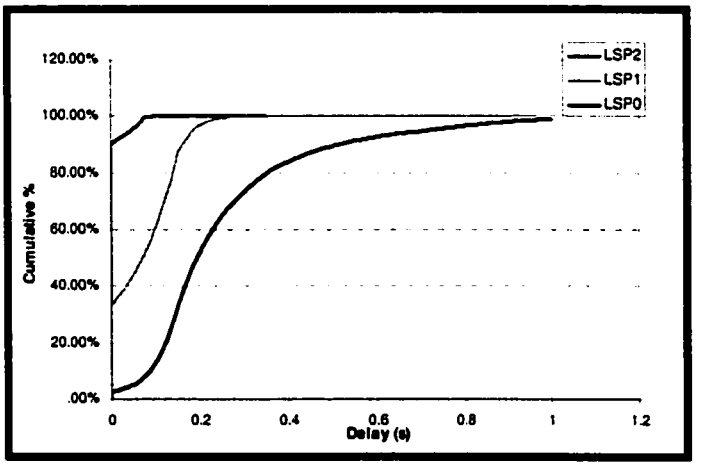

Figure B. 4 Probe Packet CDF measured one-way delay on LSPb, LSPs, LSPg

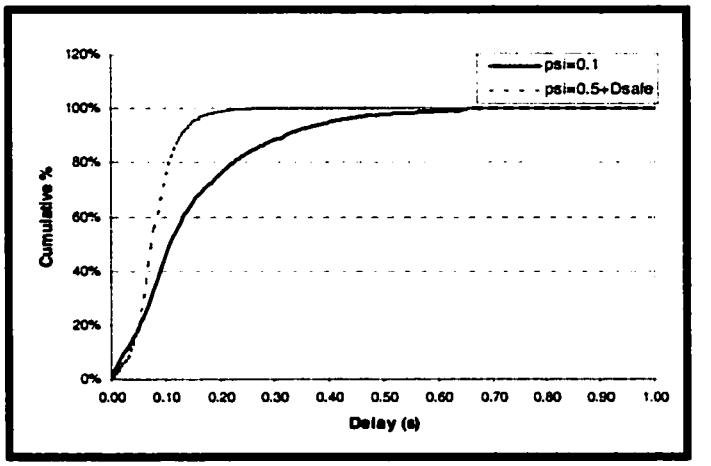

Figure B. 5 Customer (A) EF packet one-way packet delay Dth=120 ms (before and after modifying the estimator)

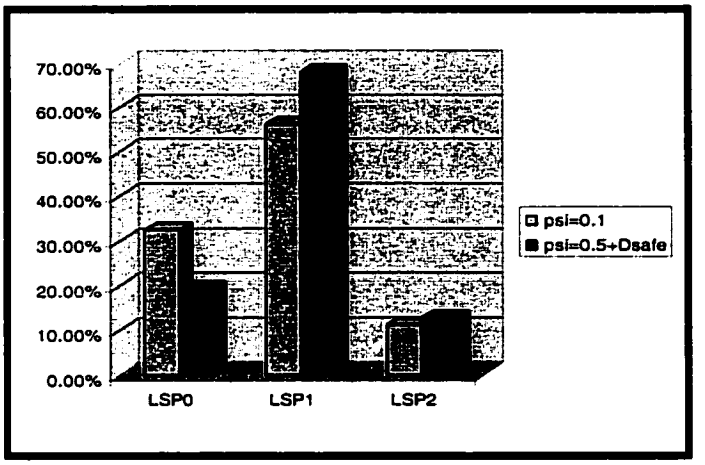

Figure B. 6 Customer A (EF) traffic distribution on each of LSP0, 1, 2 TTs, Dth= 120 ms (before and after modifying estimator)

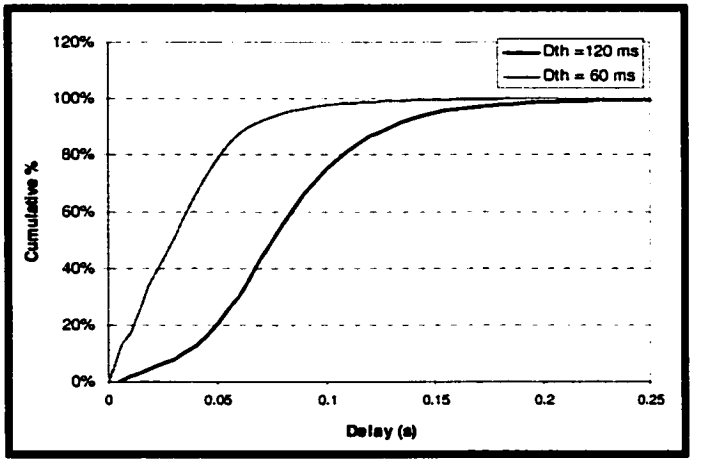

Figure B. 7 Customer (A) EF packet one-way packet delay: case 1: Dth= 60 ms, and case 2: Dth= 120 ms