Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Yu Xiao JIA

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M. A. Sc. (Electrical Engineering)

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Dynamic Quality of Service Support in Virtual Private Networks

D. Makrakis

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

V. Groza	I. Lambadaris

J.-M. De Koninck, Ph.D

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

DYNAMIC QUALITY OF SERVICE SUPPORT IN VIRTUAL PRIVATE NETWORKS

YUXIAO JIA

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

School of Information Technology and Engineering
Ottawa-Carleton Institute for Electrical and Computer Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 0-494-01502-0
Our file Notre référence
ISBN: 0-494-01502-0

ABSTRACT

This thesis presents a framework for the provision of dynamic Quality of Service (QoS) support in Virtual Private Networks (VPNs). Specifically, we focus on MPLS based VPN, with the QoS components including use of MPLS Diffserv, MPLS Traffic Engineering, RSVP-TE signaling protocol etc. The framework consists of a Dynamic Bandwidth Allocation model that includes traffic estimators and the development of a resource reservation technique.

The Dynamic Bandwidth Allocation (DBA) uses traffic estimators whose outputs are used for the reservation of resource within some time duration in the future. Three traffic estimation algorithms are implemented and tested. The Dynamic Classed Based Queuing (CBQ) technique is applied in our systems to allocate resources. This model can automatically adjust to the bandwidth size of a VPN tunnel based on how much traffic is flowing through the tunnel. An Internet Service Provider (ISP) can simplify the task of managing its network and reduce costs by using our DBA mechanism, taking advantage of the available tunnel bandwidth while still providing guarantees for high-priority traffic.

We implemented and evaluated this model on our MPLS Diffserv enabled Linux test-bed. Performance evaluation shows a higher resource utilization can be achieved by using our model.

ACKNOWLEDGEMENTS

First, I would start by thanking my supervisors Prof. Dimitrios Makrakis for his attention, technical assistance and advice whenever I need them, and most importantly for putting faith in me to conduct this work.

I would like to thank my family whose faith in me has provided—and still does—the strength to carry on. Special appreciation goes out to my parents and my brother for their encouragement and assistance when I was in difficult situation.

Last but not the least I would like to thank all my friends at the Broadband Wireless and Internetworking Research Lab in the University of Ottawa, especially Miguel Lopez-Guerrero for their friendship, help and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASE	Non-Gaussian α–stable Estimator
CBQ	Class-Based Queuing
CBR	Constraint-Based Routing
CR-LDP	Constraint-based Routed Label Distribution Protocol
Diffserv	Differentiated Services
DSCP	Differentiated Services Code Point
ECN	Early Congestion Notification
E-LSP	Exp-Inferred-PHB Scheduling Class (PSC) LSP
FIFO	First In First Out
FTN	Forwarding to NHLFE
GE	Gaussian Estimator
HTB	Hierarchical Token Bucket
IETF	Internet Engineering Task Force
Intserv	Integrated Service
IP	Internet Protocol
IPDV	Instantaneous Packet Delay Variation
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
L_LSP	Label-Inferred- PHB Scheduling Class (PSC) LSP
LRD	Long Range Dependent
LSP	Label Switched Path
MATE	MPLS Adaptive Traffic Engineering
ME	Maximum Estimator
MPEG	Moving Picture Experts Group
MPLS	Multi-Protocol Label Switching
NHLFE	Next Hop Label Forwarding Entry

OSPF	Open Shortest Path First
PHB	Per Hop Behavior
PQ	Priority Queuing
PSC	PHB Scheduling Class
QoS	Quality of Service
RED	Random Early Detection
RIP	Routing Information Protocol
RSVP	Resource Reservation Protocol
RSVP_TE	Resource Reservation Protocol –Traffic Engineering
SLA	Service Level Agreement
SP	Service Provider
SRD	Short Range Dependent
TBF	Token Bucket Filter
TCP	Transmission Control Protocol
TOS	Type Of Service
UDP	User Datagram Protocol
VPN	Virtual Private Network

Chapter 1

INTRODUCTION

1.1 BACKGROUND

Nothing has changed the modern world as rapidly as Internet [1]. Today, the Internet has become one of the most important carriers of information. People have started using the Internet to receive education, shop for groceries and do banking and stock transactions. As Internet becomes more and more important, the requirements placed on Internet become increasingly higher as well. Driven by the need for Quality of Service (QoS) support and security, the Internet technologies have evolved rapidly over the past several years [2][3].

Currently, the most common approach used by Internet Service Providers (ISPs) for the provision of QoS is to over-provision and over-engineer their operational networks. This is an expensive and inefficient solution, since it can take up to six months from planning the upgrade, to adding new fiber and/or equipment and having the operation of the entire network fully tested. In addition, some of the links are at most 10% or 20% utilized due to this conservative approach. On the other hand, the end-to-end support is achieved by concatenation of bilateral peering agreements with neighboring ISPs. Such agreements describe the volume of traffic exchanged between the two networks, but the offered performance assurance is very vague and usually not verified.

Virtual Private Networks (VPNs) [4], as an alternative to private leased lines, connect private corporate networks over the shared public Internet infrastructure. Corporations have turned to VPN solutions in order to build a secure Wide Area Network (WAN) that can deliver performance and manageability to their various sites scattered around the country. VPN customers require, at a minimum, an estimable performance over VPN's secure tunnels, which is usually specified in the Service Level Agreements (SLAs). A SLA is a contract between the service provider and the customer that specifies the maximum packet transmission rate promised by the customer, and the desired service level in terms of delay, throughput, and loss characteristics. SLAs have traditionally focused on backbone performance, such as backbone network availability, maximum or average delay, and mean time to notify customers of a network outage. Unfortunately, such coarse-grained guarantees do not reflect end-to-end performance of individual applications.

In recent years, there has been considerable research focused on extending the Internet architecture to provide better QoS support [2][3]. Two service models that have been proposed by the IETF are: Integrated Services (Intserv) [9][10][11] and Differentiated Services (Diffserv) [14][15].

Intserv with RSVP signaling [12] provides end-to-end per-flow reservations, such that each flow is guaranteed a certain amount of bandwidth at each router along its path from the source to the destination. However, this approach requires maintenance of individual states in the routers, and its signaling complexity grows with the number of users. As a result, such architecture may not scale well.

Differentiated Services (Diffserv), on the other hand, relies on packet marking and policing at the access or edge routers and different Per-Hop Behaviors (PHB) [18][19] at core routers to provide service differentiation to aggregate traffic. Edge routers are boundary points at which a flow enters or leaves a Diffserv domain, while core routers are internal routers within the domain. Edge routers may need to modify individual packets to ensure backward compatibility with external networks that do not support Diffserv. There have been comprehensive studies on the Diffserv packet forwarding mechanisms, such as scheduling, shaping, queue management, etc., but the understanding of the control framework at the session level is still relatively limited.

Since Intserv/Diffserv cannot solve all problems, additional technologies are needed. Traffic Engineering is an iterative process of network planning and network optimization [34]. The purpose of Traffic Engineering is to optimize resource efficiency and network performance. Traffic Engineering is important for providing QoS in the Internet, because

Diffserv essentially provides differentiated performance degradation for different classes of traffic during network congestion. When there is no congestion, network performance will be good even without Diffserv. Traffic Engineering is also needed in order to prevent concentration of high priority traffic. For example, if there is congestion within a certain area of the networks, Traffic Engineering will allow redirect the traffic and alleviate the bottleneck by looking at the network's condition in a more "global" manner. On the contrary Diffserv is not able to do so, since it maintains the connectionless nature of the original Internet concepts. Therefore, the contracted quality of high priority traffic may be violated [3]. Constraint-based routing, Multi-Protocol Label Switching (MPLS), an enhanced link state IGP and a path signaling protocol (e.g. RSVP-TE) are useful tools for Traffic Engineering [40][41].

The emergence of MPLS with its efficient support of explicit routing provides basic mechanisms for facilitating Traffic Engineering [40][41][42]. Explicit routing allows a particular packet stream to follow a pre-determined path rather than a path computed by hop-by-hop destination based routing such as Open Shortest Path First (OSPF) [50] or Intermediate System to Intermediate System (IS-IS) [51]. With destination-based routing as in traditional IP networks, explicit routing can only be provided by attaching to each packet the network-layer address of each node along the explicit path. However, this approach generally makes the overhead in the packet prohibitively expensive. In MPLS, a path known as a Label Switched Path (LSP) is identified by a concatenation of labels, which are stored in the nodes. As in traditional virtual-circuit packet switching, a packet is forwarded along the LSP by swapping labels. Thus, support of explicit routing in MPLS does not entail additional packet header overhead.

In summary, the provision of end-to-end QoS in IP networks has been extensively researched over the last few years. The merging of MPLS and Diffserv provides the opportunity to take advantage of the benefits provided by both of these architectures.

1.2 MOTIVATION

Historically, private WANs were provisioned using dedicated leased line connections, each line providing a point-to-point connection between two customer sites. Such networks are expensive to put in place, especially if the connections between sites need to support some level of redundancy. There is also no scope in such a system to share under-utilized bandwidth across several customers or, conversely, to increase the bandwidth available between particular sites dynamically in order to meet short-term peaks in demand.

As we discussed before, VPNs [4] are a method of interconnecting multiple sites belonging to a customer using an ISP backbone network in place of dedicated leased lines. The ISP can offer a VPN service more economically than dedicated private WANs, because the ISP provides an environment where the same backbone network resources (bandwidth, redundant links) are shared between many customers. The customer also gains by outsourcing the complex task of planning, provisioning and managing a geographically distributed network to the ISP.

VPNs services have been offered in various forms over an extended period of time and typically have been implemented at the data link layer using Frame Relay and Asynchronous Transfer Mode (ATM) networking technologies [5]. Now VPN services based on IP/MPLS and the use of the Internet are quickly gaining public interest and market acceptance. VPNs are evolving from voice to data services and from wire-line to wireless data networks. Like traditional VPNs, IP/MPLS VPNs [6][7][8] utilize shared facilities to emulate private networks and deliver reliable, secure services to end-users. Currently, most work on VPNs has mainly dealt with the security issue. However, customers of VPNs usually not only demand security, but also guaranteed QoS, in pursuance of an end-to-end service like virtual leased line. QoS enabled VPNs are highly demanded, but how to provision services dynamically on request is still a challenge for the ISP [44][46][47[48][49]. Fortunately, the advent of MPLS, DiffServ, Traffic Engineering and extended RSVP signaling protocol for Traffic Engineering (RSVP-TE) technologies make it possible to realize such services.

1.3 OBJECTIVE

The objective of this thesis is to develop a dynamic QoS support technique for VPN over heterogeneous networks. Specifically, we focus on MPLS based VPN, with the QoS components including use of MPLS Diffserv, Traffic Engineering techniques, RSVP-TE etc. A dynamic resource management model was proposed and evaluated on our Linux test-bed.

Further, we set up a MPLS Diffserv enabled Linux test-bed. Three traffic estimators and the dynamic Class Bases Queuing (CBQ) scheme are examined [52][53].

1.4 SUMMARY OF CONTRIBUTIONS

The specific contributions made through this thesis are:

- Deployment of an experimental MPLS Diffserv enabled Linux test-bed, which is able to apply dynamic resource allocation in real time. In our implementation, the RSVP-TE daemon running under each MPLS router is responsible for the RSVP-TE signaling and the maintenance of the MPLS state, e.g. allocation and installation of the MPLS labels during LSP set up and freeing and removing labels on LSP tear down. The Dynamic Class Based Queuing (CBQ) script runs on the each node to dynamically reallocate resources based on the estimate of future demands.

- Implementation of the three traffic estimation algorithms, namely Maximum Estimator (**ME**) [46][99], Gaussian Estimator (**GE**) [46] and non-Gaussian α–stable Estimator (**ASE**) [114].

- Development of a α–stable long-range dependent traffic generator, which produces traffic with α–stable statistics. This allows us to emulate realistically aggregate traffic of video, ftp, www and other modern applications running through Internet [59][60]. Using α–stable traffic as an example of traffic load, we performed experiments for the evaluation of our proposed architecture and resource allocation mechanism. We also used other traffic loads, such as UDP Poisson traffic.

- Conducting of thorough performance evaluation, and comparison of these competing technologies. We have run extensive experiments to explore the effectiveness of our estimators.

1.5 THESIS ORGANIZATION

The remainder of this thesis is organized as follows. Chapter 2 provides an overview of the VPN technology and the related QoS support technologies. The VPN technology is addressed first. Then two Internet service models, the Intserv/RSVP, and Diffserv model are discussed. MPLS and Traffic Engineering as a complementary technology are described as well. Finally we consider the Diffserv over MPLS, an attractive strategy for QoS support and an attractive solution for deployment of the ISPs' network. Related work addressing the QoS control in VPN is surveyed.

Chapter 3 presents our research methodology and explains how our dynamic bandwidth allocation approach works. We first describe the general framework. Following three traffic estimation algorithms have been presented. Finally the dynamic resource reservation technology is discussed.

Chapter 4 describes our Linux experimental test-bed configuration. The Linux traffic control mechanism and MPLS Diffserv support under Linux are reviewed. We also discuss the network performance tools (HW/SW) and traffic sources used in our work.

Chapter 5 presents the performance evaluation results and the related performance analysis that is based on the acquired results.

Chapter 6 concludes the thesis and discusses directions for future work.

Chapter 2

QOS SUPPORTING TECHNOLOGIES IN THE INTERNET

The service quality of a network is reflected by the extent of user satisfaction of the network. Although Quality of Service (QoS) is dependent on the application type and user perception, the emphasis of QoS in this thesis is at the network level, which can usually be measured by packet delay, jitter, packet loss rate and application throughput [2]. Currently Internet only provides best effort service. With the rapid transformation of the Internet into a commercial infrastructure, the need for QoS has rapidly appeared.

In this chapter, we describe several QoS supporting technologies in the Internet. The VPN service, which provides a cost efficient alternative of the expensive leased line service becomes adopted by more companies and ISPs. Two traditional service models, developed by the Internet Engineering Task Force (IETF) for the provision of QoS in Internet, are introduced. They are the Integrated Service (Intserv) with resource reservation protocol (RSVP) and Differentiated Service (Diffserv). Since Intserv/Diffserv cannot solve all the problems, additional technologies are needed. MPLS comes into the picture by providing a fast-forwarding mechanism. Traffic Engineering works to facilitate efficient and reliable network operations while simultaneously optimizing network resource utilization and traffic performance. Finally, we discuss the integration approach of Diffserv over MPLS. Related work on QoS in VPN is presented as well.

2.1 VPN TECHNOLOGY OVERVIEW

A Virtual Private Network (VPN) [4] is a private network connecting distant locations of an organization, by utilizing resources of the public network infrastructure. This solution yields substantial cost savings since it obviates the use of dedicated links by using resources from the public infrastructure. In the roughly ten years since their emergence, the VPNs technology typically has been implemented at the data link layer using Frame Relay and ATM networking technologies [5]. Now VPN services based on IP/MPLS [4][6][7][8] and the use of the Internet are quickly gaining public interest and market acceptance. VPNs are evolving from voice to data services and from wireline to wireless data networks. Like traditional VPNs, IP/MPLS VPNs utilize shared facilities to emulate private networks and deliver reliable, secure services to end-users.

VPN service models can be classified as:

- Traditional VPNs
  - Frame Relay (Layer 2)
  - ATM (Layer 2)
- CPE-based VPNs (Customer Premises Equipment)
  - L2TP and PPTP (Layer 2)
  - IPSec (Layer 3)
- Provider Provisioned VPNs (PP-VPNs)
  - MPLS-based Layer 2 VPNs
  - BGP/MPLS VPNs (Layer 3)
  - IP-based VPN (Layer 3)

Presently, most of the work in VPNs is addressing security issues. However, our growing dependence and use of real-time, interactive and other QoS sensitive applications in our personal lives, demand that effective QoS capabilities are included in VPN technology. Nevertheless, provision of QoS introduces new challenges to the existing technology. First of all, it will require introduction of new features in the routers. Second, many customers may not have the knowledge and resources to deploy and manage enhanced Internet services by themselves. They would rather outsource the complex task to their Internet Service Provider (ISP). The ISP can provide VPN services more economically, by sharing the resources of the common network backbone.

This thesis focuses on providing a QoS enabled MPLS VPN solution. It builds on the attractive features of MPLS [7][8]. IETF has proposed several service models and protocols for providing QoS in the Internet. Notably among them are the Intserv/RSVP, Diffserv, MPLS and Traffic Engineering. They are briefly reviewed in the subsequent sections.

2.2 INTEGRATED SERVICES AND RSVP

The Integrated Service (Intserv) model [9] proposed two service classes in addition to best effort service. They are: guaranteed service [10] for applications requiring fixed delay bound; controlled load service [11] for applications requiring reliable and enhanced best effort service. The philosophy of this model is that there is an inescapable requirement for routers to be able to reserve resources in order to provide special QoS for specific user packet streams, or flows. This in turn requires flow-specific state in the routers [9].

RSVP was developed as a signaling protocol to reserve resources [12]. The signaling process is illustrated in Figure 2-1. The sender sends a PATH message to the receiver specifying the characteristics of the traffic. Every intermediate router along the path forwards the PATH message to the next hop determined by the routing protocol. Upon receiving a PATH message, the receiver responds with a RESV message to request resources for the flow. Every intermediate router along the path can reject or accept the request of the RESV message. If the request is rejected, the router will send an error message to the receiver, and the signaling process will terminate. If the request is accepted, link bandwidth and buffer space are allocated for the flow and the related flow state information will be installed in the router.

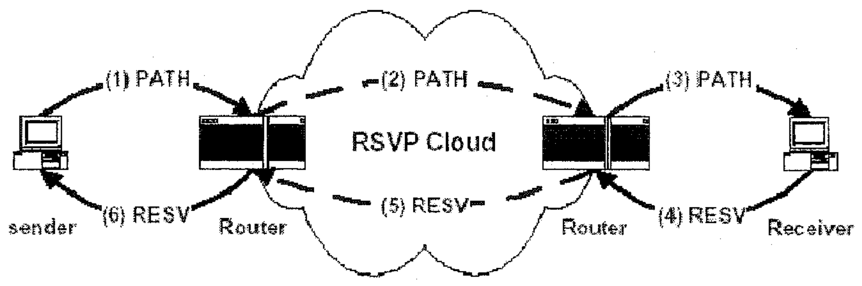

Figure 2-1. RSVP signaling

Intserv is implemented by four components: the signaling protocol (e.g. RSVP), the admission control routine, the classifier and the packet scheduler. Applications requiring guaranteed service or controlled-load service must set up the paths and reserve resources before transmitting their data. The admission control routines will decide whether a request for resources can be granted. When a router receives a packet, the classifier will perform a multi-field (MF) classification and put the packet in a specific queue based on the classification result. The packet scheduler will then schedule the packet accordingly to meet its QoS requirements.

Problems with the Intserv architecture are: 1) the amount of state information increases proportionally with the number of flows. This places a huge storage and processing overhead on the backbone routers. Therefore, this architecture does not scale well in the Internet backbone; 2) the requirement on routers is high. All routers must support RSVP, admission control, MF classification and packet scheduling; 3) ubiquitous deployment is required for guaranteed service. Incremental deployment of controlled load service is possible by deploying controlled-load service and RSVP functionality at the bottleneck nodes of a domain and tunneling the RSVP messages over other parts of the domain [13].

2.3 DIFFERENTIATED SERVICES

Because of the shortcoming in implementing and deploying Intserv and RSVP, Differentiated Services (Diffserv) [14][15] is defined. The essence of Diffserv is to divide traffic into multiple classes, and treat them differently, especially when there is a shortage of resources.

The IPv4 header contains a Type of Service (ToS) part explained in [16]. Diffserv renames the ToS octet of IP header as Differentiated Services field (DS field) [17] and uses it to indicate the forwarding treatment a packet should receive. The six most significant bits of DS 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 (Figure 2-2).

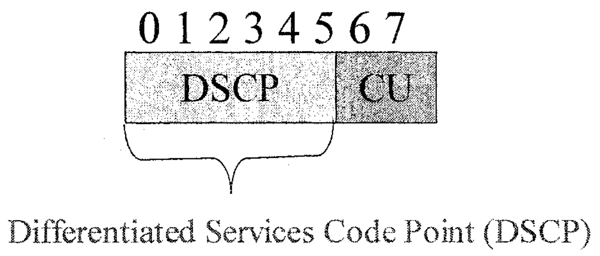

Figure 2-2. Diffserv field

Diffserv also standardizes a number of Per-Hop Behavior (PHB) groups [18][19]. Using different classification, policing, shaping and scheduling rules, several classes of service can be provided.

The Diffserv mechanism has packets be classified, policed, marked and possibly shaped at the ISP edge routers, besides queued and scheduled at the all routers (edge and core). Traffic maybe shaped again before being sent out to another domain, or remarked after arriving to another domain. An illustration of Diffserv functionality at the edge and core router is shown in Figure 2-3. These building functions will be discussed in the following subsections.

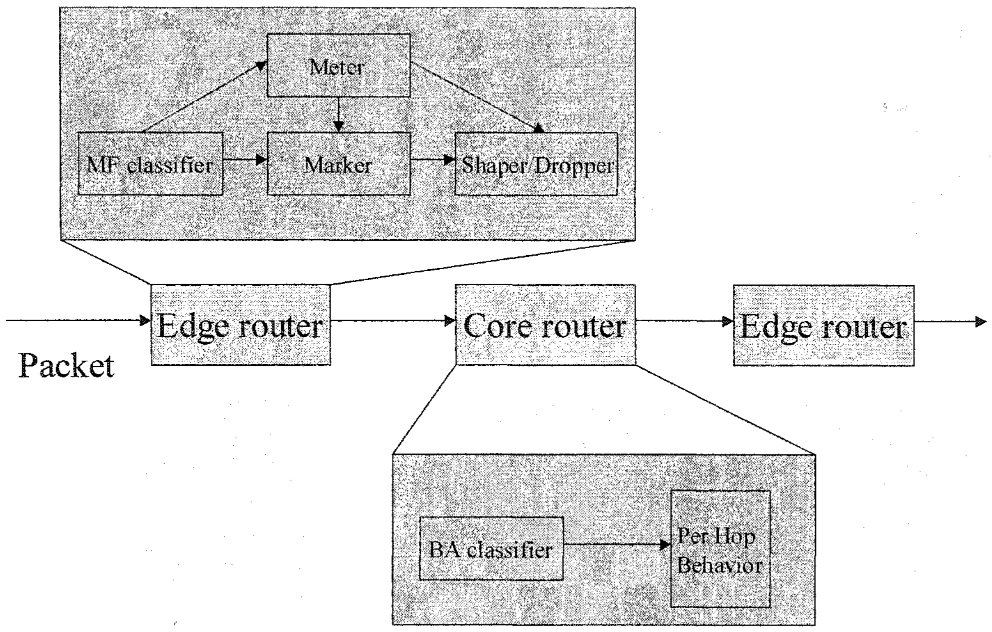

Figure 2-3. An illustration of Diffserv functionality at boundary node and interior node

2.3.1 Service Classes

The Diffserv architecture defines three classes of service: best effort, assured service and premium service. Best effort is the traditional Internet service.

Assured service provides reliable and estimable packet delivery. It is intended for non-real time interactive applications such as Web browsing. Traffic in the assured class will exhibit the Assured Forwarding (AF) Per-Hop Behavior (PHB) [19]. The AF class has four subclasses, AF1-AF4, each with its own forwarding priority and three levels of drop precedence. Customers that need assured service should have Service Level Agreements (SLAs) with their ISPs. The SLAs will specify the amount of bandwidth allocated for the customers. Customers are responsible for deciding how their applications share that amount of bandwidth. SLAs for assured service are usually static, i.e., customers can start data transmission whenever they want without signaling to their ISPs.

Premium service provides reliable, low-delay and low-jitter service, suitable for Internet telephony, video conferencing, or for creating virtual leased lines for the support of VPNs. Traffic in premium class will exhibit the Expedited Forwarding (EF) PHB [18]. Each customer that subscribes to the premium service will have a SLA with its ISP. The SLA specifies a desired peak bit-rate for a specific flow or an aggregation of flows [3]. The customer is responsible for not exceeding the peak rate. Otherwise, excess traffic may be dropped. Alternatively, if an ISP is certain that there are enough resources to carry the extra premium traffic, the extra premium traffic can still be transmitted. The customer will be charged accordingly. Because ISP networks are usually over-provisioned, the second case is more likely. Since premium service is more expensive than assured service, it is desirable for ISPs to support both static SLAs and dynamic SLAs [66][67][98]. Dynamic SLAs allow customers to request premium service on demand without subscribing to it. Signaling and admission control are needed for dynamic SLAs.

2.3.2 Classification, Policing, Marking, Shaping at the Ingress Router

Classification is mainly based on the incoming interface. This incoming interface can be physical or logical. Besides the incoming interface, source and destination IP addresses, source and destination port numbers, protocol ID and the ToS octet can be further used in the classification. This is called multi-field (MF) classification. By considering the incoming interface first, the classification rules for traffic from one interface can be separated from rules for traffic from other interfaces. This greatly reduces the number of rules that need to be considered, making the MF classification much faster. Based on the classification result, traffic profiles and corresponding policing, marking and shaping rules can be applied.

Policing is to determine whether traffic from a particular customer conforms to the traffic profile specified in the SLA, and take action accordingly. 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.

Marking is the mechanism setting the DSCPs of the packets. Core routers identify the classification of packets based on the DSCP field in the IP header, and use it to select a PHB behavior for that packet.

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. From the technical point, shaping and policing functions are complementary.

2.3.3 Queue Management and Packet Scheduling at the Ingress and Core Router

After a packet is marked, its treatment at edge and core routers is solely determined by its DSCP. The decision on how to choose and process a packet is done according to the queuing discipline followed by the network node.

It is quite evident that queuing discipline is directly linked to the ability of providing QoS and determines what are the QoS characteristics the node and in extension the network can provide.

Each network device must implement some queuing discipline that governs how packets are buffered while waiting to be transmitted. The queuing discipline can be thought of as allocating both bandwidth and buffer space. Currently, there are 12 types of queuing disciplines supported in Linux kernel 2.4, Hierarchical Token Bucket (HTB) was added in Linux kernel 2.4.20. Since our experimental test-bed is implemented on the Linux framework, we will introduce several types of traffic forwarding discipline that are commonly used in our test bed.

2.3.3.1 First In First Out (FIFO)

First In First Out (FIFO) queuing [20] is the most basic queue scheduling discipline. In FIFO queuing, all packets are treated equally by placing them into a single queue, and then servicing them in the same order that they were placed into the queue. Packets are queued until the buffer space has been used; new arrivals finding the buffer full are dropped. As the packet processing is based only on the order of packet arrival, FIFO does not discriminate packets coming from different traffic sources, nor it provides differential treatment. FIFO is the default queuing discipline used by Linux and most of the (best effort) routers around the world. There are two formats for FIFO: PFIFO and BFIFO [81]. PFIFO constrains the queue size as measured in packets. BFIFO does so as measured in bytes.

Priority Queuing (PQ) [20] can be regarded as an extension version of FIFO. It is an easy way of providing differential treatment to packets and one of the first queue scheduling algorithms implemented to support differentiated services. In classic PQ, packets are first classified by the system and then placed into different priority queues. Packets are scheduled from the head of a given queue only if all queues of higher priority are empty. Within each of the priority queues, packets are scheduled in FIFO order. It is evident that in PQ, classes of low priority could end up in starvation (experiencing service denial) if the network node has to move through considerable volume of traffic associated with the high priority classes.

FIFO and PQ are very simple to implement. FIFO is used in the "best effort" based Internet. Also, PQ was used in most first generation networking products that had some traffic differentiation capabilities incorporated in their structure. The price paid for this simplicity of implementation is somewhat reduced capability of QoS provision.

2.3.3.2 Token Bucket Filter (TBF)

The Token Bucket Filter (TBF) [22] is a simple queue that only passes packets arriving at a rate, which is not exceeding some administratively set rate, with the possibility to allow short bursts in excess of his rate. TBF is very precise, network- and processor friendly. It can be the first choice if you simply want to slow an interface down. The TBF implementation consists of a buffer (bucket), constantly filled by some virtual pieces of information called tokens, at a specific rate (token rate). The most important parameter of the bucket is its size, that is the number of tokens it can store. In order to send a packet, the sender must hold a token. If no token is available, the packet is either marked as low priority or is dropped.

2.3.3.3 Random Early Detection (RED)

The RED queuing discipline [21] is a congestion avoidance and congestion control algorithm. It reacts to congestion either by dropping packets arriving at the router or by downgrading their classification (by changing the value of the appropriate bit in the packet header). The algorithm detects congestion by computing the average queue size. The average queue size is calculated by using a low-pass filter with an exponential weighted moving average (EWMA). The average queue size is compared with two thresholds: a minimum threshold ($\min_{th}$) and a maximum threshold ($\max_{th}$). When the average queue size is less than $\min_{th}$, no packets are dropped. When the average queue size is greater than $\max_{th}$, each arriving packet is dropped. This ensures that the average queue size does not significantly exceed the maximum threshold. When the average queue size is between $\min_{th}$ and $\max_{th}$, each arriving packet is dropped with probability P_a, where P_a is an increasing function of the average queue size. The probability that a packet is dropped from a particular connection is roughly proportional to that connection's share of the bandwidth at the gateway.

Since RED can accommodate transient congestion by controlling the average queue size, RED is good to provide high throughput and low average delay in high-speed networks with TCP connections that have large windows.

2.3.3.4 Class Based Queuing (CBQ)

A: <u>CBQ Overview</u>

CBQ is the most important queuing algorithm used in our implementation. The main idea of CBQ algorithm [23][24][25][26][27][29][30][31] is hierarchical link sharing and resource management. Figure 2-4 demonstrates the link-sharing concept of CBQ. The link-sharing goals are defined the following:

- Each interior or leaf class should receive roughly its allocated link-sharing bandwidth over appropriate time intervals in times of congestion.

- If some leaf or interior classes are not using their allocated bandwidth, the distribution of excess bandwidth among other classes should not be arbitrarily done; it should be distributed in a controlled and fair manner.

Table 2-1 shows the terminologies frequently used in CBQ. There are three different proposals to implement link-sharing guidelines [23]: Formal, Ancestors-Only and Top-Level. Ancestors-Only and Top-Level are approximations of Formal link-sharing guidelines. The most robust and easy to implement is the Top-Level link-sharing proposals, whose guidelines are as follows:

A class continues unregulated if one of the following conditions holds:

- The class is not over-limit, OR
- The class has an under-limit ancestor whose level is at most Top-Level.

Otherwise, The link-sharing scheduler will regulate the class.

These link-sharing guidelines provide a good compromise between the scheduler implementation complexity and its performance as far as satisfying the link-sharing goals. More details regarding the implementation of CBQ can be found in [23] [24].

Figure 2-5 demonstrates the CBQ building block [30]. Incoming traffic is put into the appropriate queue according to a set of filtering rules applied by the classifier. The general scheduler extracts packets from queues and it guarantees each class to receive at least its nominal bandwidth. The estimator measures the inter-packet departure time for each class and checks whether the class is exceeding its allocated rate (over-limit class). The link-sharing scheduler cooperates with this "feedback block" and distributes the excess bandwidth according to the link-sharing structure.

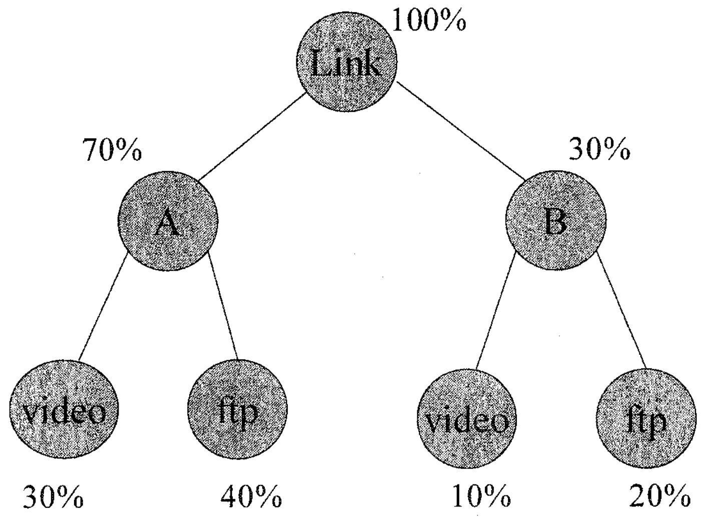

Figure 2-4. An example of hierarchical link-sharing structure

Table 2-1. Terminology in CBQ

Over-limit, under-limit, at-limit	If a class has recently used more than its allocated bandwidth, it is called over-limit. If less, it is called under-limit. Otherwise, at-limit.
Link-sharing scheduler	It is used for leaf classes that have exceeded their link-sharing allocations during congestion period, distributes the excess bandwidth according to the link-sharing structure.
General scheduler	It schedules packets without any consideration of the link-sharing guidelines; it is used in periods of non-congestion. It extracts packets from queues and guarantees each class to receive at least its nominal bandwidth. It is proposed to use WRR (Weighted Round Robin) with weights proportional to allocated bandwidth.
Regulated, unregulated	A class is unregulated if it is being scheduled by the general scheduler. If it is being scheduled by the link-sharing scheduler, it is called regulated.
Satisfied, unsatisfied	For a leaf class, if it's under-limit and has a persistent backlog, it is unsatisfied. For an interior class, if it is under-limit and has some descendant class with a persistent backlog, it is unsatisfied. Otherwise, the class is satisfied.
Formal link-sharing guidelines	Original link-sharing guidelines, most complex.
Top-level link-sharing guidelines	An approximation of the computing of formal link-sharing guidelines, more robust than other approximation.

Ancestor-only link-sharing guidelines	Another approximation of the computing of formal link-sharing guidelines.
Classifier	Examines each packet and puts it into the queue based on the IP address, port and transport layer protocol.
Estimator	It measures the inter-packet departure time for each class and checks whether the class is exceeding its allocated rate (using EWMA).
Bounded, isolated	A class is marked as bounded if it is not allowed to borrow. It is marked as isolated if it does not allow non-descendant classes to borrow its unused bandwidth and it also does not borrow from others.
Level	A leaf class is at level 1; the level increases upward.

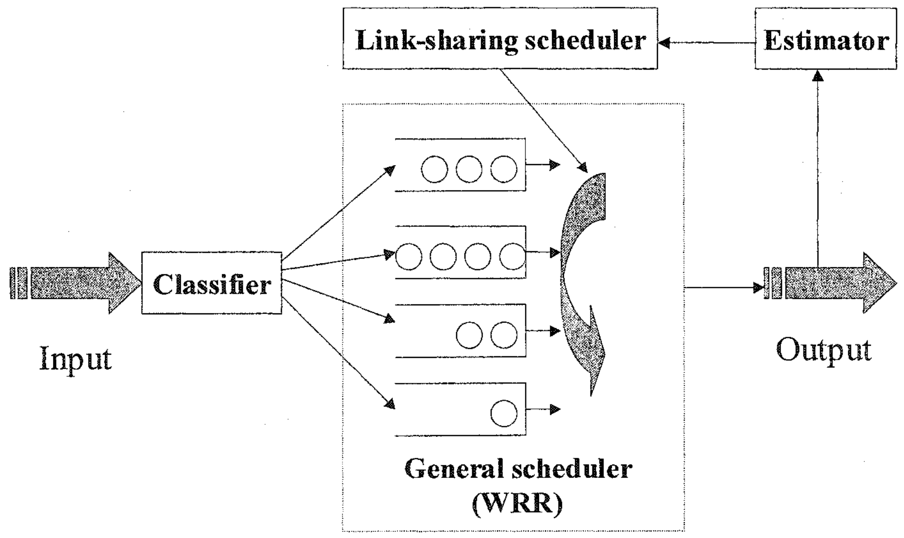

Figure 2-5. CBQ building blocks

B: CBQ Parameters

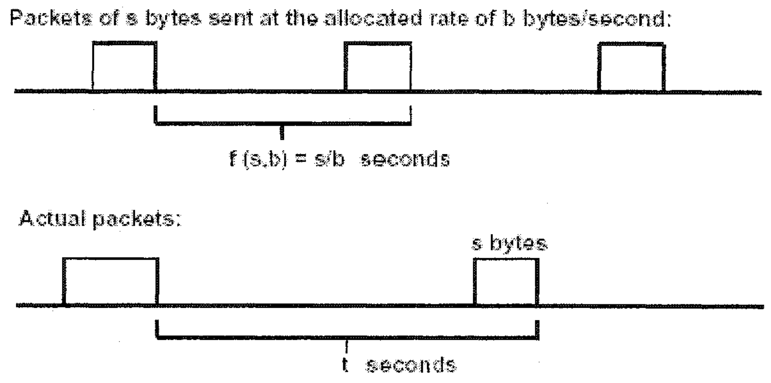
Packets of s bytes sent at the allocated rate of b bytes/second:
f (s,b) = s/b seconds
Actual packets:
s bytes
t seconds

Figure 2-6. "idle" variable computation for the limit status of a class

Here is the summary of parameters defined in CBQ algorithm.

- *s*: the size of recently transmitted packet in bytes
- *b*: the link-sharing bandwidth allocated to the class in bytes per second
- *l*: the total link bandwidth (bytes/sec)
- *t*: the actual measured inter-departure time between two successive packets
- *w*: weight in EWMA (exponential weighted moving average)
- *maxburst*: *maxidle* is derived from this. (M)
- *minburst*: steady-state burst size. (m)

CBQ uses several parameters to control the output pattern [30]. *Idle* is computed by difference from actual inter-departure time (t) and the ideal one (s/b) (see Equation (2.1) and Figure 2-6). *Idle* takes into account whether the class is over-limit or under-limit. However, the most important variable is *avgidle,* which keeps track of the actual rate of the traffic. It is calculated by EWMA (exponential weighted moving average) function (see Equation (2.2)). If *avgidle* <= 0, it means over-limit, otherwise, it is under-limit. Each class is suspended for a certain amount of time (*extradelay*) as soon as *avgidle* becomes negative. This forces the class rate to be compliant with the allocated bandwidth. *Extradelay* is calculated in order to allow the class sustaining a steady state burst of m packets after the suspension (see Equation (2.3)).

$$idle = t - s / b \qquad (2.1)$$

$$avgidle_{n+1} = (1-w) * avgidle_n + w * idle \qquad (2.2)$$

$$extradelay = (s/_b - s/_l) * \left(1 + \frac{1-(1-w)^{m-1}}{w*(1-w)^{m-1}}\right) \qquad (2.3)$$

$$\max idle = (s/_b - s/_l) * \left(\frac{1}{(1-w)^{M-1}} - 1\right) \qquad (2.4)$$

Variables *minidle* and *maxidle* define a minimum and maximum bound to *avgidle*. *Minidle* usually are set to zero [25]. *Maxidle* sets a maximum value for *avgidle* to avoid that a previously idle class could be allowed sending for too much time unpunished. The equation for *maxidle* is shown in (2.4). In CBQ implementation, *maxidle* and *extradelay* are not set directly because of their small degree of intuitiveness. These parameters are substituted by *maxburst* (M) and *minburst* (m), which stand for the maximum number of back-to-back packets that the class is allowed to send after a long idle period and the maximum number of back-to-back packets that the class is allowed to send during steady state.

Since CBQ uses this idle based estimator, CBQ is not always working well. Two problems can be foreseen in the idle based estimator. One is that it is hard to implement without having precise clock or without knowing the hardware speed of the network device (i.e. variable rate modems or wireless adapters). The other is that it is hard to interpret idle estimator parameters. This is why a new scheduling algorithm called Hierarchichal Token Bucket (HTB) [28][32] comes to the picture. We will describe HTB algorithm later.

C: CBQ Limitation

As we discussed above, the idea behind CBQ is good, but in a real life situation, it is not working very well. When CBQ calculates the idle time of the link, it depends on many unknown parameters, i.e. size of the packets, physical link bandwidth, other bottlenecks in the network etc. All of these result in the limitation of CBQ in Linux [28].

- Accuracy of rate control: CBQ has error margins of several percent against the REAL interface speed. We will explore this limitation in our Linux implementation.

- Difficulty to set for slow interfaces: There have been some difficulties to set the right parameters when the link is slow (i.e. less than 512 Kbps). Also, it is better to assign more than 10% of the link bandwidth to each class. Setting high-priority to an interactive class could improve the response time. In our Linux implementation, we deploy small traffic rate due to the limitation of hardware. This will reduce the system performance relatively.

- Using a fine-grained kernel timer: CBQ shapes the outgoing traffic using the kernel timer. It is better to use a fine-grained kernel timer when use CBQ especially on Fast Ethernet. We set our kernel tick as high-resolution value in order to increase the accuracy of our experiments.

2.3.3.5 Hierarchical Token Bucket (HTB)

HTB is a classful qdisc with a simpler set of configuration parameters than CBQ. The objectives is precision and ease of implementation. Both CBQ and HTB can help you to control the use of the outbound bandwidth on a given link. Both allow you to use one physical link to simulate several slower links and to send different kinds of traffic on different simulated links. In both cases, you have to specify how to divide the physical link into simulated links and how to decide which simulated link to use for a given packet to be sent. The key difference between HTB and CBQ is that HTB uses TBF as the rate estimator, which is simple to setup and implement. This also makes HTB more precise. CBQ uses the idle based estimator as we discussed before.

Conceptually, HTB is an arbitrary number of token buckets arranged in a hierarchy. The algorithm of HTB used for packets schedule is following: first select all leaves classes whose rate is not reached and send packets according to the priority. For leaves with same priority, DRR (Deficit Round Robin) [33] is used. When rates for all leaves are exceeded, it will borrow bandwidth from parents or grandparents.

Three parameters need to be set in HTB: *rate*, *ceil* and *burst*. *rate* means how much the guaranteed bandwidth is available for a given class. *ceil* is short for ceiling, which indicates the maximum bandwidth that a class is allowed to consume. *Burst* is the amount of data that can be sent in a single session at the maximum speed for a single class. Any bandwidth used between rate and ceil is borrowed from a parent class. A number of children classes can be made under root class, each of which can be allocated some amount of the available bandwidth from the parent class. In these children classes, the *rate* and *ceil* parameter values need not be the same as suggested for the parent class. This allows you to reserve a specified amount of bandwidth for a particular class. It also allows HTB to calculate the ratio of distribution of available bandwidth to the ratios of the classes themselves.

2.3.4 How Diffserv Apply to VPN

We described earlier the Diffserv and VPN techniques separately. However, Diffserv can provide QoS to Internet VPNs because both technologies fit well with each other by following commonalities [44]:

- Both architectures logically separate service traffic from public traffic.
- Both are enforced in edge routers.
- Both aggregate traffic flows.
- Both classify IP packets.

For providing QoS guarantees similar to those customers who are used to from leased line services, the Expedited Forwarding (EF) Per-Hop Behavior (PHB) [18] seems to be the appropriate choice. The EF PHB can be used to build a low-latency assured-bandwidth end-to-end service through Diffserv domains. Such a service appears to the endpoints like a point-to-point connection or virtual leased line. A typical SLA for such a service might include the ingress and egress points of the Diffserv domain that shall provide the service and a peak-rate, which can be guaranteed to the traffic stream.

The Assured Forwarding PHB group [19] is a way to offer different levels of forwarding assurances for IP packets received from a customer Diffserv domain. Four AF classes are defined with each having three drop precedences. AF is considered more complex to configure in Diffserv domains. However, we also see great potential in AF for VPNs. For example, a customer might prefer to specify a bandwidth range [Y, Z] rather than a single peak rate for a VPN tunnel. Y can be configured as the rate for low drop precedence and Z-Y as the rate for medium drop precedence.

When providing Diffserv-to-VPN tunnels, special attention must be given to DSCP mapping at the tunnel starting point. In outsourced VPNs, the ingress router of an ISP might perform Diffserv classification and traffic conditioning as well as tunnel encapsulation. If Diffserv processing is done first, the DSCP of the inner IP header must be copied to the DSCP field of the outer IP header; if encapsulation is performed first, the ingress router can select the appropriate DSCP for the corresponding tunnel.

2.4 TRAFFIC ENGINEERING

A major goal of Internet Traffic Engineering is to facilitate efficient and reliable network operations while simultaneously optimizing network resource utilization and performance [34]. Traffic Engineering has become an indispensable function in many large Autonomous Systems (AS) because of the high cost of network assets and the commercial and competitive nature of the Internet. These factors emphasize the need for maximal operational efficiency.

The key performance objectives associated with Traffic Engineering can be classified as being either:

- Traffic oriented or
- Resource oriented

Traffic oriented performance objectives indicate the aspects that enhance the QoS of traffic streams. For example, in a single class (best effort Internet service model), the key traffic oriented performance objectives include: minimization of packet loss, minimization of delay, maximization of throughput, and enforcement of service level agreements. Resource oriented performance objectives indicate the aspects pertaining to the optimization of resource utilization. Efficient management of network resources is the vehicle for the attainment of resource oriented performance objectives. In particular, it is recommended to ensure that subsets of network resources do not become over utilized and congested while other subsets along alternate feasible paths remain underutilized. Bandwidth is a crucial resource in contemporary networks. Therefore, a central function of Traffic Engineering is to efficiently manage bandwidth resources.

Minimizing congestion is a primary traffic and resource oriented performance task. Our interest here is on congestion problems that are prolonged rather than on transient congestion resulting from instantaneous bursts. Congestion typically happens two scenarios:

- When network resources are insufficient or inadequate to accommodate offered load.

- When traffic streams are inefficiently mapped onto available resources, causing subsets of network resources to become over-utilized while others remain underutilized.

The first type of congestion problem can be addressed by either, (i) expansion of capacity, (ii) application of classical congestion control techniques, or (iii) both. The second type of congestion problems, namely those resulting from inefficient resource allocation, can usually be addressed through Traffic Engineering.

In general, congestion resulting from inefficient resource allocation can be reduced by adopting some load balancing policies. The objective of such strategies is to minimize maximum congestion or alternatively to minimize maximum resource utilization, through efficient resource allocation. If congestion is minimized through efficient resource allocation, packet loss decreases, transit delay decreases, and aggregate throughput increases. Thereby, the perception of network service quality experienced by end users becomes significantly enhanced.

Clearly, load balancing is an important network performance optimization policy. Nevertheless, the capabilities provided for Traffic Engineering should be flexible enough so that network administrators can implement other policies, which take into account the prevailing cost structure and the utility or revenue model.

In order to implement traffic engineering effectively, the IETF introduced MPLS [35], constraint-based routing and enhanced link state IGPs. Here MPLS technology will be reviewed since it is the key technology used in our work.

2.4.1 Traffic Engineering for VPN

As we describe above, Traffic Engineering is the process of controlling how traffic flows through a network in order to optimize resource utilization and network performance [34]. The basic motivation for Traffic Engineering comes from the fact that shortest-path-based routing leads to congestion on certain links while others remain relatively unused. This result in inefficient network resource usage and also increases the probability that a VPN tunnel cannot be established due to lacking resources along the shortest path between tunnel ingress and egress points. Traffic Engineering in the past has been constructed on overlay models, running IP over an underlying connection-oriented network technology such as ATM or frame relay [44]. However, there are several problems with the overlay model. Most important is the management complexity with handling two kinds of network technology — IP and the underlying network (ATM or frame relay). Furthermore, mesh-like networks do not scale for large VPNs. MPLS can overcome several limitation of the overlay model. We will describe its attractive features in Section 2.5.

2.5 MULTI-PROTOCOL LABEL SWITCHING (MPLS)

MPLS is an advanced data-forwarding scheme. It extends routing with respect to packet forwarding and path controlling [35]. MPLS has emerged as the preferred technology for providing QoS, traffic engineering and Virtual Private Network (VPN) capabilities over the Internet.

2.5.1 MPLS Architecture

Each MPLS packet has a header (see Figure 2-7). In a non-ATM environment, the header contains a 20-bit label, a 3-bit experimental field (formerly known as class of service, or CoS, field), a 1-bit label stack indicator and an 8-bit TTL field. In an ATM environment, the header contains only a label encoded in the VCI/VPI field. An MPLS capable router, termed label-switching router (LSR), examines the label and possibly the experimental field and uses this information to forward the packet.

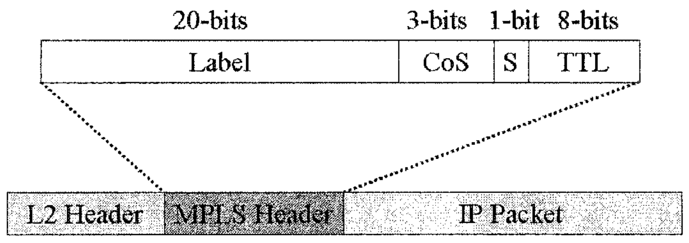

Figure 2-7. MPLS header architecture

At the ingress LSRs of an MPLS-capable domain IP packets are classified and routed based on a combination of the information carried in the IP header of the packets and the local routing information maintained by the LSRs. An MPLS header is then inserted for each packet. Within an MPLS-capable domain, each LSR will use the label as the index to look up the forwarding table of the LSR. The packet is processed as specified by the forwarding table entry. The incoming label is replaced by the outgoing label and the packet is switched to the next LSR. This label-switching process is very similar to ATM's VCI/VPI processing. Before a packet leaves a MPLS domain, its MPLS header is removed. This whole process is showed in Figure 2-8. The paths between the ingress LSRs and the egress LSRs are called Label Switched Paths (LSPs). MPLS uses some signaling protocol like Label Distribution Protocol (LDP) [36], Resource ReSerVation Protocol – Traffic Engineering (RSVP-TE) [37][38], or Constraint-based Routed Label Distribution Protocol (CR-LDP) [36][39] to set up LSPs.

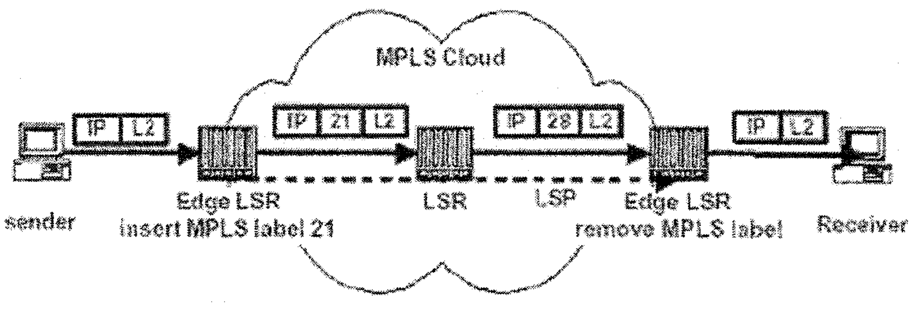

Figure 2-8. MPLS forwarding diagram

2.5.2 MPLS Signaling Protocol (RSVP-TE)

RSVP-TE and CR-LDP are defined to provide support for Traffic Engineering. Although the two protocols provide a similar level of service, the way they operate is different, and the detailed functionality they offer is also not consistent. Hardware vendors and network providers need clear information to help them decide which protocol to implement in a Traffic Engineered MPLS network. Each protocol has its champions and detractors. A detailed comparison between these two protocols can be found in [39].

Currently it appears that RSVP-TE dominates the market. In our Linux test-bed implementation, RSVP-TE was used. Thus, we describe this protocol in more detail.

The use of RSVP-TE as a signaling protocol for Traffic Engineering is quite different than what it was envisioned by its original developers in the mid-1990s. Figure 2-9 shows the difference between traditional RSVP and RSVP-TE.

RSVP-TE signaling takes place between pairs of routers (rather than pairs of hosts) that act as the ingress and egress points of a traffic trunk. It installs state that applies to a collection of flows that share a common path and a common pool of shared network resources, rather than a single host-to-host flow. By aggregating numerous host-to-host flows into each LSP tunnel, RSVP-TE significantly reduces the amount of RSVP state that needs to be maintained in the core of a service provider's network. RSVP-TE signaling installs distributed state related to packet forwarding, including the distribution of MPLS labels. The path established by RSVP-TE signaling is not constrained by conventional destination-based routing, so it is the perfect tool to establish Traffic Engineering trunks.

To establish an LSP tunnel, the ingress LSR transmits an RSVP PATH message downstream to the egress LSR. The egress LSR responds to the receipt of the RSVP PATH message by transmitting an RSVP RESV message upstream toward the ingress LSR. When the ingress LSR receives the RSVP RESV message, the LSP is established and the ingress LSR can use the LSP tunnel to forward traffic to the egress LSR. The basic flow for setting up an LSP using RSVP-TE for LSP Tunnels is shown in Figure 2-10 below.

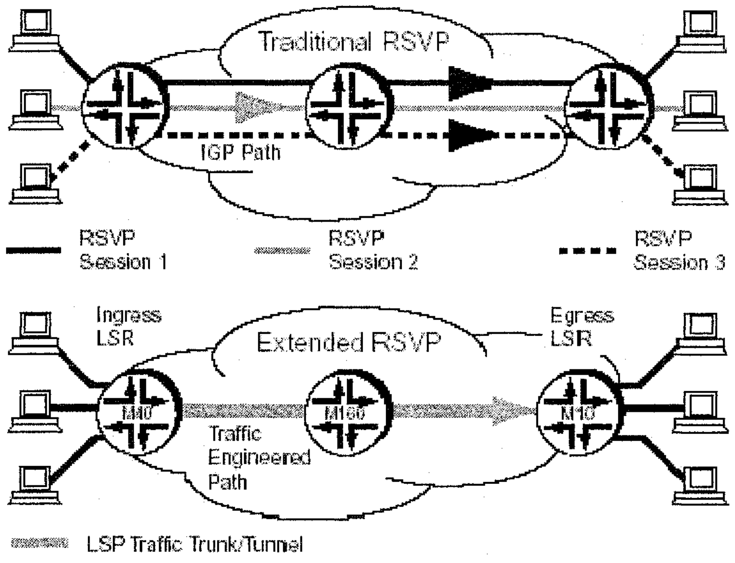

Figure 2-9. Traditional RSVP and RSVP-TE

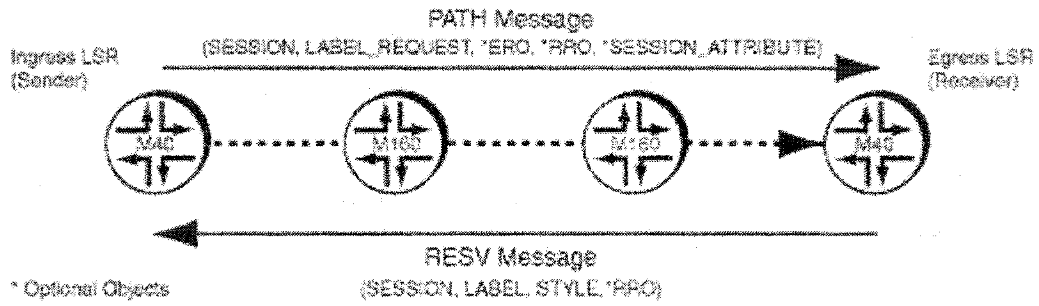

Figure 2-10. RSVP-TE PATH message and RECV message

2.5.3 MPLS Traffic Engineering

MPLS is strategically significant for Traffic Engineering because it can potentially provide most of the functionality available from the overlay model [40][41][42], in an integrated manner, and at a lower cost than the currently competing alternatives.

The attractiveness of MPLS for Traffic Engineering has already been addressed in the past few years. Many of the existing proposals for Traffic Engineering over MPLS focus only on the potential to create explicit LSPs. Although this capability is fundamental for Traffic Engineering, it is not really sufficient. Additional augmentations are required to foster the actualization of policies leading to performance optimization of large operational networks.

Anwar Elwalid et.all [43] proposed a MPLS adaptive traffic engineering mechanism called MATE. The main goal of MATE is to avoid network congestion by adaptively balancing the load among multiple paths based on measurement and analysis of path congestion. Esmael Dinan et all [70] investigate the use of a dynamic traffic partitioning and assignment methodology to adaptively map ingress traffic into several parallel Label Switched Paths (LSPs) in MPLS based IP networks. Tarek Saad proposes an alternate LSP Cheapest Path First algorithm (CPF) [71] that attempts to forward the time-critical traffic onto the cheaper path as long as it meets the QoS constraints of the time-critical traffic. The most interesting part is that they use delay estimation algorithm [72][90][100][73][74] to measure the path performance.

2.5.4 MPLS VPN

Here we summarize the features of MPLS for VPN [7][8][45]:

- MPLS offers fast forwarding capability.
- MPLS connects sites through setting up label switch paths (LSPs) on which traffic engineering can be applied.
- MPLS provides support for various L2 protocols, e.g. ATM, Frame Relay etc.

- MPLS supports signaling protocols (RSVP-TE, CR-LDP), which can facilitate fast configurations of VPNs.

- MPLS is capable of scaling into very large networks

There is two main features that make MPLS fit for use in VPNs: tunneling and label stacking. MPLS allows tunnels to be set up by appending a MPLS header in front of the IP header. This 32-bit MPLS header avoids the large overhead of another IP header required with IP-in-IP tunneling. The MPLS header can therefore be used several times (i.e., labels can be stacked).

Label stacking is being used in particular when building MPLS/Border Gateway Protocol (BGP)-based VPNs [8]. In such a case, a packet is classified at an ingress router of an ISP based on the interface belonging to a particular VPN. The ingress router has learned via BGP to which VPN it belongs, to which egress router the packet must be sent, and via which egress interface the destination is reachable. The ingress router appends two labels to a packet belonging to a VPN. The inner label specifies the egress port at the ISPs egress router (i.e., the link toward the destination sub-network of the VPN). The outer label is being used to forward the packet toward the egress router and can be learned by MPLS signaling protocols such as CR-LDP or RSVP-TE. Both labels are popped by the egress router. Note that MPLS makes the private VPN addresses of a customer transparent to the routers of the ISP (tunneling).

Many VPN customers want to receive guaranteed minimum bandwidth on their VPN connections, but it would be inefficient and costly, for both the ISP and their customers, to provision fixed bandwidth LSP tunnels that could support the maximum bandwidth needed between all VPN sites. A far better option is to provision the networks with spare capacity over and above the minimum bandwidth requirements and to share the spare capacity in the network between the VPN customers and public Internet traffic.

Customers also expect VPN data to remain private, including the topology and addressing scheme for their network as well as the data carried on the VPN. Historically, VPN implementations based on ATM [5] or Frame Relay VCs have provided this security by virtue of the connection-oriented nature of the physical network. However, the connectionless public IP network cannot provide the same protection, and IP VPNs have relied on cryptographic means to provide security and authentication.

MPLS brings to IP security benefits similar to layer 2 VCs. This means that the customer equipment connected to the VPN does not need to run IPSec [56] or other cryptographic software, representing a considerable saving for the customer in terms of equipment expense and management complexity.

MPLS VPN security is achieved as described below.

- At the ISP edge router, all data for a VPN is assigned a label stack that is unique to the VPN destination. This ensures that the data is delivered only to that destination, so data does not leak out of the VPN.

- Any other packet entering the ISP network is either routed without the use of MPLS or is assigned a different label stack, thus a malicious third-party cannot insert data into the VPN from outside the SP network.

- ISP routers can use the Cryptographic Algorithm MD5 [57], or similar techniques, to protect against insertion of fake labels or LSRs into the label distribution protocols.

2.6 DIFFERENTIATED SERVICES OVER MPLS

Diffserv is a scalable service architecture, which provides scalable QoS guarantees to aggregated traffic. MPLS is a Layer 3 switching technology that provides fast packet forwarding and traffic engineering. They are similar in some sense. In both domains, packet are marked (DSCP) or labeled at edges. Intermediate nodes look at these values (DSCP or Label) and take appropriate actions. Thus, the combination of Diffserv and MPLS presents a very attractive strategy to backbone network service providers with scalable QoS and traffic engineering capabilities [58][61].

To integrate Diffserv with MPLS, one needs a way to map the DSCP values in Diffserv to the shim header in MPLS, since the DSCP field is not directly visible to MPLS Label Switch Routers (LSRs) that forward packets based on the MPLS header. However, the EXP field is 3 bits long, while DSCP field is 6 bits. This requires that multiple DSCP classes are mapped to three EXP bits. There are two methods defined by IETF to deal with this problem: Exp-Inferred-PHB Scheduling Class (PSC) LSP (E-LSP) and Label-Inferred-PSC LSP (L-LSP) [58].

E-LSP determines the PHB of a packet solely from the EXP field and can support up to Eight PHB per E-LSP. The EXP field conveys the queuing, scheduling and drop precedence to the LSR. 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.

L-LSP determines the PHB of a packet from both the label and EXP fields. The label field determines the PSC (queuing and scheduling) while the EXP field determines the PHB (drop precedence). An arbitrarily large number of PHBs can be supported. Some signaling protocol such as RSVP-TE, CR-LDP can be used to support PHB signaling during L-LSP establishment. E-LSP scales better than L-LSP because it results in less state information and signaling operations to be handled by the LSRs in the MPLS network. Figure 2-11 illustrates the mapping from DSCP to EXP at the ingress node.

In a Diffserv supported MPLS network, the following three stages are needed:

a) At the Ingress - Mapping the DSCP of an incoming packet to the shim header of the outgoing packet, the shim header is pushed on top of the IP packet.

b) At the Core (LSRs) - Mapping Diffserv indication from an incoming labeled packet to the outgoing labeled packet. Usually the EXP is copied from the top label of an incoming packet to top label of the outgoing packet.

c) At the Egress - Mapping of Diffserv requirements (EXP-value and/or label-value) from an incoming labeled packet to the DSCP value of the outgoing IP packet; the bottom label is popped.

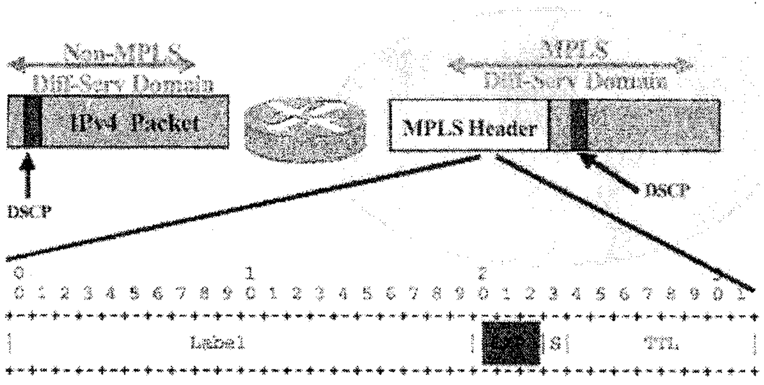

Figure 2-11. An illustration of mapping from DSCP field to Label/EXP field

2.7 RELATED WORK

There has been a wide spectrum of work during the last few years on Quality of Service (QoS) control and management in VPNs. Here we present a survey of related work that serves as background to our research.

N. G. Duffield et.al [46] proposed a flexible service model for resource management in VPNs, called *hose*, which specifies the capacity required for aggregate traffic from one endpoint to the set of other endpoints in the VPN customer sites. Each hose is associated with a performance guarantee. The share of resources allocated to a hose is resized based on a set of traffic estimators, and the performance of this adaptive scheme is compared to static provisioning through trace-driven simulations. The authors consider traces of telephone calls over the AT&T national long distance network as well as data traffic on a large corporate private network. Results show that the dynamic resizing method achieves a factor of 2 to 3 in capacity savings on access links over statically provisioned customer-pipes. However, this work only considers a single ISP scenario.

To extend to address inter-domain resource allocation in the case of multiple domains (ISPs), Chen-Nec Chuah et.al [47] have designed a Clearing House (CH) architecture that facilitates resource reservations over multiple network domains, and performs local admission control. Two key ideas employed in this design to make the CH scalable to a large user base are hierarchy and aggregation. In this model, they assume the network is composed of various basic routing domains, which can be aggregated to form logical domains. This introduces a hierarchical tree of logical domains and a distributed CH architecture is associated with each logical domain to maintain the intra-domain aggregate reservations. The parent CH in the logical tree maintains the inter-domain reservation requests. Call setup time is reduced by performing advanced reservations based on statistical estimates of the call traffic across various links. They explore, with simulations, the efficiency of the CH-architecture in terms of resource utilization, call rejections and reservation setup time.

Another extensive research on QoS VPNs has been done through a CATI (Charging and Accounting Technology for the Internet) project funded by the Swiss National Science Foundation (SNF) [111]. The detailed QoS support for VPN can be found in [44][48][49].

Chapter 3

DYNAMIC BANDWIDTH ALLOCATION

A large number of organizations have geographically dispersed operations with local networks, supporting the information processing needs at each branch. Typically, reliable and secure interconnection of these sites is accomplished with point-to-point dedicated communication lines, leased from a service provider. Alternatively, Virtual Private Network (VPN) technology can provide virtual dedicated lines over the Internet. This solution yields substantial cost savings, since it obviates the use of dedicated links, by using resources from the public infrastructure. Presently, most of the work in VPNs is addressing security issues. However, our growing dependence and use of real-time, interactive and other QoS sensitive applications in our personal lives, demand that effective QoS capabilities are also included in this technology.

In this Chapter, we describe our dynamic bandwidth allocation approach. We describe a general framework in Section 3.1. In our architecture, Diffserv over MPLS is used as the baseline in the pursuit for end-to-end QoS. Section 3.2 presents three traffic estimation schemes. The dynamic resource reservation technique is discussed in Section 3.3. Dynamic Class Based Queuing (CBQ) was chosen to dynamically reallocate resources based on the estimation of future demands.

3.1 GENERAL FRAMEWORK

3.1.1 Use of MPLS and Diffserv

Existing VPN technologies are based on the provision of some form of bandwidth between sites. These were based mostly on ATM and Frame Relay technologies, which allow define easily QoS specifications and allocate bandwidth. Now VPN services based on IP/MPLS and the use of the Internet are quickly gaining public interest and market acceptance. The problem with IP technology is that provision of QoS is very difficult, since it is a connectionless technology. We can provide strict QoS guarantees by using Intserv, since it allows for the definition of specific routes through the network with QoS guarantees. However, this approach does not scale well. While the number of VPNs to be deployed might not be that large at present, the target should be that in the future, this number will increase dramatically. Should a user friendly and cost efficient technology becomes possible, it will allow this happen. A possible solution is to provide services similar to those of SVC (Switched Virtual Circuit) in ATM, where users can dynamically request connections from the ATM network, pretty much the same way we are dialing a number in the PSTN (Public Switched Telephone Network). In addition to the fact we have to be looking to the future, thus look for accommodating a large number of users, there is also the issue that existing Internet is based on Diffserv, due to its better scalability. To use Intserv means we have to deploy Intserv capable routers within the infrastructure, which means cost.

Based on this analysis, it is important that an architecture based on Diffserv for QoS support is adopted. Conventional Diffserv Internet is connectionless oriented. This means that no specific path can be defined, and the composition of traffic in terms of flows passing through certain router can change (we can not specify what flows are to be passing through a certain router at certain time). This creates difficulties in providing QoS, since Diffserv provides mainly priorities than guarantees. Should we have strict control on the number of flows passing through the network at certain time, we then have a better understanding and the ability to define stricter statistical QoS guarantees for the traffic streams passing through the defined path, belonging to a certain class. In addition to this, the Diffserv model has defined certain levels of classes, but these classes are not linked with any scalability in terms of QoS performance. Take for example the classification of EF class. We have defined only one EF class, which can be used for multiple applications such as voice, interactive video, video-on-demand etc. These applications are real time but have different QoS requirements. Voice needs an end-to-end delay around 300 ms and low jitter but can sustain packet losses in the range of 1%. On the contrary, video-on-demand needs low packet loses (less than 0.01% or even less) and low jitter; however, it can sustain long end-to-end delays since it is not an interactive application. Interactive video needs the losses of video-on-demand and end-to-end delay of voice. Should we go with a completely "homogeneous" network (we design the network to meet statistically the needs of the applications), we have to do so for the worst-case scenario, which is the case of interactive video. This will produce a very conservatively designed network (for example, most of the existing traffic would be voice, however, we provide resources for video) and reduction of efficiency. By using MPLS, we can define specific paths for EF class through the network, which can be engineered for the specific applications. In addition, MPLS will allow us to control the number of flows passing through the path; thus, we are able to avoid congestions and violation of QoS requirements more successfully.

3.1.2 Architecture and Possible Scenarios

There are two different scenarios we can consider running through. In the first scenario, we can consider an ISP, providing connectivity for one or several VPNs, each requiring certain QoS. The combination of MPLS and Diffserv will allow provide the granularity required by the applications. The users might be maintaining connectivity within long periods of time, but the number of applications running through might be changing. Take for example a mining company running Tele-operation activities to a remote site. The number of video links will be changing, depending on the operations running. The number of operations might change with the time as well. It can be because of change of shifts, stopping the vehicles for maintenance, change of the operation (for example, from Tele-operating a moving vehicle to Tele-operating a static drip etc.). Since this is prime bandwidth, should the ISP be able to determine the requirements of this stream, it can use the remaining of bandwidth for other purposes. The traffic is classified as EF within the Diffserv domain, in order to be processed with full priority as compared to other classes (AF, BE).

Another scenario deals with the reservation requests of the VPN user itself. Let us go back to the case mentioned earlier, a mining company Tele-operation. There is a gateway between the company's premise and the public network. As many links can be ran between the two sides, it might be very difficult to determine the exact requirements of bandwidth by examining each connection individually. However, by estimating the aggregate traffic, the gateway of the user, estimates the bandwidth needs of the aggregate flow, and send it to the ISP, thus changing the reservations in real time, purchasing as much bandwidth as needed. The ISP, at its ingress router, will police the traffic of the VPN user, and if it goes above the requested value, it will drop the excess traffic. The policed traffic goes through, which is treated as EF. In this case, the dropping is happening at the ingress router. Figure 3-1 demonstrates this scenario.

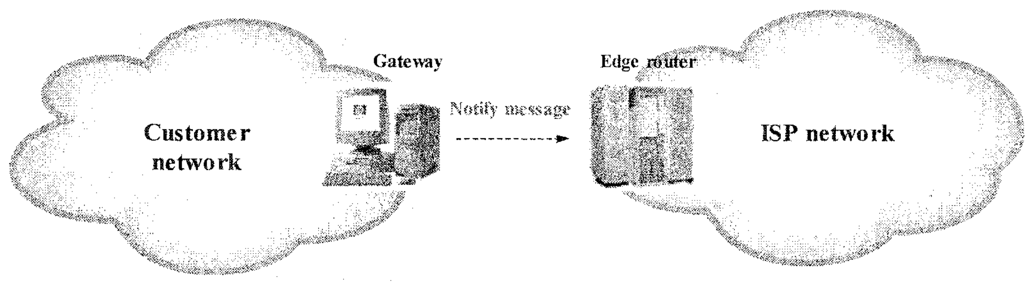

Figure 3-1. One possible scenario of our architecture

Traffic estimators running at the Gateway of customer network, predicts the bandwidth to be used in the near future. Then, the Gateway sends notify messages, which contain the estimated value for the next time period, to the Edge router of ISP network periodically. Based on the predicted value, the Edge router of the ISP network does policing for the traffic. If the real traffic rate is higher that the policing rate, the packets will be dropped or downgraded.

3.2 TRAFFIC ESTIMATORS

Here we discuss three traffic estimators, which are used to estimate the resource needs of a traffic stream. We are interested in flows comprising traffic aggregated at either the pipe, or the VPN level. We assume that the measurements comprise samples gathered at regularly spaced instants during a window of duration T_{meas}. The samples are themselves some function of the traffic rates in the interval between sampling instants. The measurements are used to estimate the bandwidth for the traffic flow over some window of duration Y, following the measurement widow. Such estimations have the locality property that they depend only on measurements over the window of duration T_{meas} into the past. The parameters, defined here:

- X: inter-sample interval (second or minute)
- Y: reservation window (second or minute)
- W: number of samples taken within the measuring window T_{meas}
- N: number of samples taken within the reservation window Y
- R_i: average sample rate over inter-sample interval X

Specific traffic estimation algorithms that we employ are: Maximum Estimator (**ME**), Gaussian Estimator (**GE**) and non-Gaussion α–stable Estimator (**ASE**). For each estimator, we apply sliding window scheme: $Y = N*X$ (N = 1,2, ……W,…..). N is parameter. Since $T_{meas} = W*X$. Either $Y \le T_{meas}$, or $Y > T_{meas}$. $Y = T_{meas}$ is a special case of this scheme. Figure 3-2 demonstrates this sliding window scheme. In our Linux implementation we test the case $Y \le T_{meas}$.

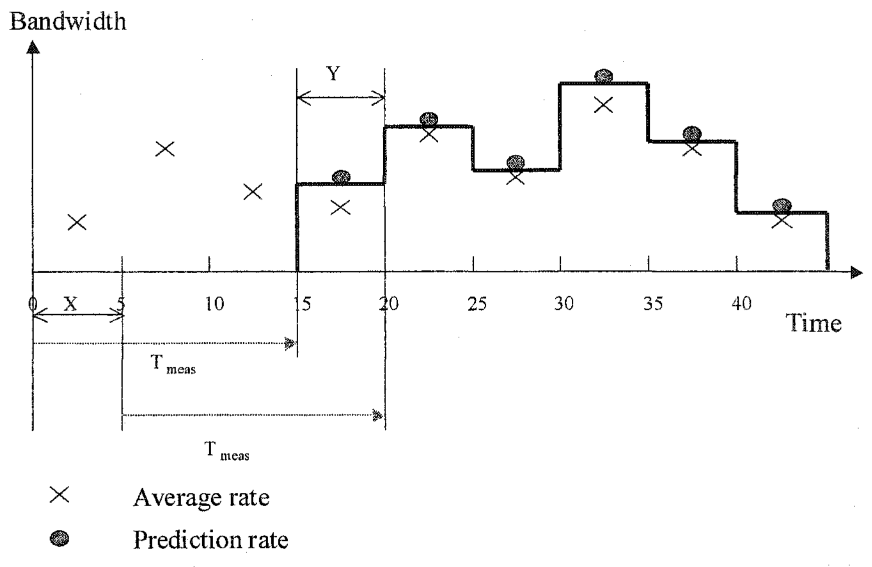

Figure 3-2. An illustration of sliding window scheme: $Y = N^*X$ (N = 1)

3.2.1 Maximum Estimator (ME)

The original algorithm can also be found in [46][99]. The renegotiated rate R_{ren} is the maximum of the rate sampled during the measurement window T_{meas}. We assume the average rate over the inter-sample interval R_i.

$$R_{ren} = \max\{R_i\} \qquad (3.1)$$

Practically, we monitor the *X* average counter, keep track of the largest value of *X* average and apply it as the expected value of traffic over some configurable interval *Y*. After the *Y* interval has expired, the largest *X* average is cleared in order to record the new largest *X* average over the next *Y* interval. How to choose *X* and *Y* is a performance trade-off. We show the effect of parameters *X*, *Y* and T_{meas} on the system's performance in Chapter 5.

We have applied a modification on the original algorithm, by adding a margin *M*.

$$R_{ren} = \max\{R_i\} * (1 + M) \qquad (3.2)$$

3.2.2 Gaussian Estimator (GE)

The algorithm is based on the assumption that aggregated traffic can be characterized by the Gaussian distribution. The renegotiated rate R_{ren} is equal to

$$R_{ren} = m + a\sqrt{v} \qquad (3.3)$$

where m and v are respectively the mean and variance of the rates R_i sampled during the measurement window, a is a scale factor that controls the extent to which the negotiated rate accommodates the variability of the samples. The interpretation of a is that in a Gaussian approximation of the rate distribution, we expect the bandwidth $m + a\sqrt{v}$ to be exceeded with probability 1-G(a), where G(.) is the cumulative distribution of the standard normal distribution [54][68]. N. G. Duffield et.all describes this algorithm in [46].

This approach is based on one fundamental concept: estimation of aggregate traffic. The capacity requirement of a trunk is easier to estimate because the statistical variability of the individual flows is smoothed by the process of aggregation. As mentioned earlier, we model the envelope of the arrival process of a traffic trunk as Gaussian distribution. When the number of individual flows gets large, and assuming they are iid (independent identically distribution) and have finite variance, according to the Central Limit Theorem [69], the aggregate arrival rate R_i tends to follow the Gaussian distribution. In a bufferless fluid model, losses occur when the total arrival rate R_i exceeds the reserved bandwidth R_{ren}. In our analysis, we consider sufficient buffer space to hold the largest packet, and approximate the probability of packet loss as p. We assume the arrival rate R_i has a Gaussian distribution. If we reserve R_{ren}, p is approximately equal to

$$p \approx p(R_i > R_{ren}) = G(a) \qquad (3.4)$$

To fulfill the QoS needs of real time applications (especially the real-time and loss sensitive ones), a should be chosen carefully so that the loss rate is within the limits specified by the application. If the Gaussian approximation holds, $a = G^{-1}(p)$. We explore how a different margin size affects the system's performance by using different values of a.

Table 3-1 is a simplified table to calculate a In order to keep thing simple, we use the parameter k in our implementation. k is defined as $k = -\log_{10}\{p\}$, thus, when we choose k = 1, it means the packet loss rate is 10% and the corresponding value for a is a =1.2815.

Table 3-1. Probability of packet loss p and scale factor a

k	p=10^{-k}	$a = G^{-1}(p)$
1	0.1	1.2815
2	0.01	2.3263
3	0.001	3.0902
4	0.0001	3.7190
5	0.00001	4.2649
6	0.000001	4.7535

However, Equation (3.3) assumes no queue present. An arriving packet has to find the server free and be processed immediately, or else will be dropped. When buffer exists, the packet loss will be lower than p.

For performance evaluation refer to Chapter 5.

3.2.3 Non-Gaussian α–stable Estimator (ASE)

This estimator assumes that the statistical behavior of aggregate traffic streams is statistically described through α-stable long-range dependent stochastic processes [59][60]. This is a valid assumption, since the authors of [59][60] proved that such distributions describe more accurately aggregate traffic passing through modern networks as compared to earlier models.

Assuming a traffic model based on α-stable self-similar stochastic processes, Miguel Lopez Guerrero [114] proposes an extension of the concept of probabilistic envelope processes to the α-stable case, previously defined for more traditional models. He represented the bandwidth demand imposed by the traffic with an envelope process and showed that with this approach we can simply and effectively deal with much of the argued complexity encountered in α-stable models and develop techniques for proper dimensioning of network elements. Based on the α-stable traffic model [60], the following envelope process $\hat{W}_j$ is proposed:

$$\hat{W}_j = m + K\sigma_W \qquad (3.5)$$

where parameter m is the mean of the number of arrivals per unit time, and σ_W is the scale parameter. It is similar to the variance of the Gaussian distribution and there is formula $\sigma_W{}^2 = 1/2v$ in that Gaussian case (α = 2) [114]. How to compute σ_W can be found in [115]. The parameter K is determined by the overshoot probability ε and index of stability α according to the following equation :

$$P\{W_j > \hat{W}_j\} = P\{W_j > m + K\sigma_W\} = P\left\{\frac{W_j - m}{\sigma_W} > K\right\} = \varepsilon \qquad (3.6)$$

Table 3-2 and Table 3-3 shows how to determine K analytically from the probability ε when α = 1.95 and α = 1.60. Figure 3-3 demonstrates the α–stable traffic and envelope processes (α=1.95).

Table 3-2. Probabilities $\varepsilon = P\{X>K\}$ and resulting values of K when α=1.95

ε	0.1	0.05	0.03	0.01	0.005	0.003	0.001
K	1.82	2.36	2.72	3.45	3.94	4.35	5.83

Table 3-3. Probabilities $\varepsilon = P\{X>K\}$ and resulting values of K when α=1.60

ε	0.1	0.05	0.03	0.01	0.005	0.003	0.001
K	1.98	2.79	3.54	6.16	9.11	12.03	23.93

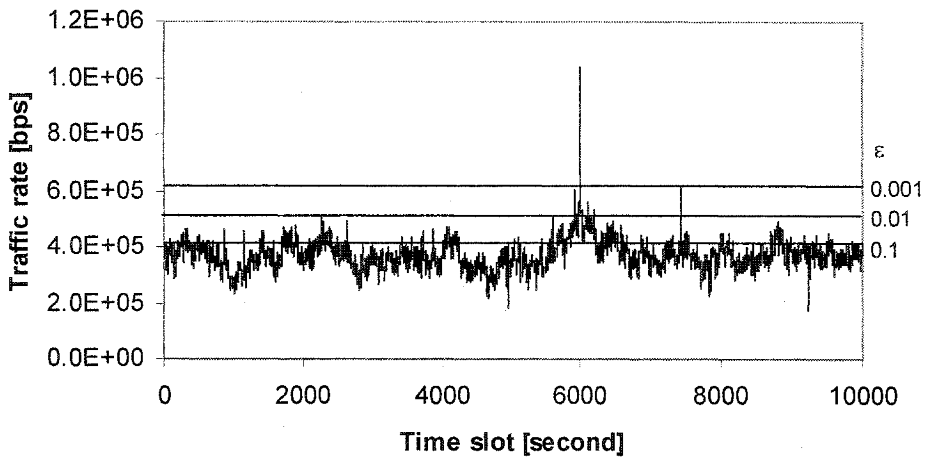

Figure 3-3. Demonstration of α–stable traffic and envelope processes

Again, Equation (3.5) assumes no buffer existing. But in reality each router has buffers inside. When taking into consideration the buffer size, the formula has to be changed [115]. Thus, the expression provided in Equation (3.5) represent an upper limit in terms of packet loss. In order to keep the things simple, we still implement the system based on the Equation (3.5). The performance evaluation can be seen in Chapter 5.

3.3 DYNAMIC RESOURCE ALLOCATION

Many VPN customers want to receive guaranteed bandwidth on their connections. However, provision of fixed bandwidth might be inefficient and costly. Excessive cost might prevent a large number of mediums to small size businesses and individuals from joining the service. A far better option is to dynamically assign the resources, based on the forecasted demand from on-line measurements. The provider may use dynamic resource allocation algorithms to pace and forward traffic according to a Service Level Agreement (SLA) established with the customer. The SLA defines the resources and QoS requirements [66][67].

The basic idea behind the estimation-based dynamic resource allocation scheme is to apply estimation techniques on traffic volumes, in order to allocate network resources to a traffic stream in future time slots. This way, we can dynamically change the amount of resource reservation for a VPN. The main benefit of dynamically resized VPNs is that the bandwidth from temporally low demanding streams can be made available to other participating connections, resulting in higher overall network utilization.

Two elements are required to implement the dynamic resource allocation scheme: a traffic estimation algorithm and a resource reservation technique [46][52]. For the former we propose to apply our three estimation algorithms as mentioned earlier. For the later two approaches are possible: dynamic CBQ and RSVP-TE. As a first step, we propose to use Dynamic Class Based Queue (CBQ). Dynamic CBQ is a *tc* script, which is used to dynamically reallocate resources based on the estimation of future demands. RSVP-TE will be considered for future work.

Internet data traffic exhibits burstiness at multiple time-scales. Therefore, an estimator based on a given sampling window can underestimate the bandwidth requirement that varies at shorter time-scales, resulting in possible violations of QoS guarantees. In our design, the edge router (ER) monitors and estimates the bandwidth requirements for the next time slot based on rates sampled during a specific measurement window T_{meas}. At the end of estimation, the ER send regular updates to the nodes along a specific link. These nodes reserve the bandwidth based on the estimated value. If the estimated bandwidth overestimates the actual bandwidth required, it will result in inefficient resource utilization. The performance heavily depends on the choice of T_{meas}, and the time-scale at which the traffic demand varies. We explore these tradeoffs in our experimental study in Chapter 5.

Our approach can be easily extended to scenario 2 as discussed on Section 3.1.2. The customer gateway monitors and estimates the bandwidth requirements for the next time slot. Then the gateway sends this message to the ER of the ISP network, the ER does policing.

Chapter 4

EXPERIMENTAL TEST-BED

In this Chapter, we first describe our experimental test-bed. Then, we explore the Linux QoS support functions. Section 4.3 shows our network performance tools consisting of hardware based GN Nettest's IW95000 [82] and software based Ethereal/Tethereal [75] and Tcpdump [76]. Finally the traffic sources used in our experimental work are discussed.

4.1 MPLS DIFFSERV NETWORK CONFIGURATION

In this section, we will describe our Linux experimental test-bed configuration. Since the Edge router plays a key role in our Diffserv network, we discuss its functionality in detail. Some common used Linux tool terminologies are listed as well.

4.1.1 Experimental Test-bed

We implemented an experimental MPLS Diffserv enabled Linux test-bed based on the open source project "RSVP-TE daemon for Diffserv over MPLS under Linux" [55]. Figure 4-1 shows the network architecture of our test-bed. It consists of two customer networks and one MPLS backbone network. The source customer network includes an IP router used to mark IP packets (setting up the DSCP value) by using "iptables" [101][102]. "iptables" is the part of framework inside the Linux 2.4.x kernel which enables packet filtering, network address translation (NAT) and packet mangling. Specifically, it is a generic table structure for the definition of rule sets. Each rule within an IP table consists of a number of classifiers and one connected action. The MPLS backbone network is made up of two edge LERs and one core LSR, which establish one Diffserv enabled LSP. All routers run under the Linux Redhat^{TM} system. The kernel version for our application is 2.4.19. Background traffic generation and network monitoring can be done using two Inter-Watch 95000 (IW95000) network analysis/traffic generation tools, manufactured by GN Nettest [82]. In order to capture the MPLS packets, we also employ the software based Ethereal/Tethereal packet sniffer application [75] supporting MPLS decoders — an updated version of the Tcpdump [76] application that runs on Windows and Linux. The network capacity of the core LSR is 10 Mbps, which is used to create congestion. The other segments are configured with 100 Mbps capacity.

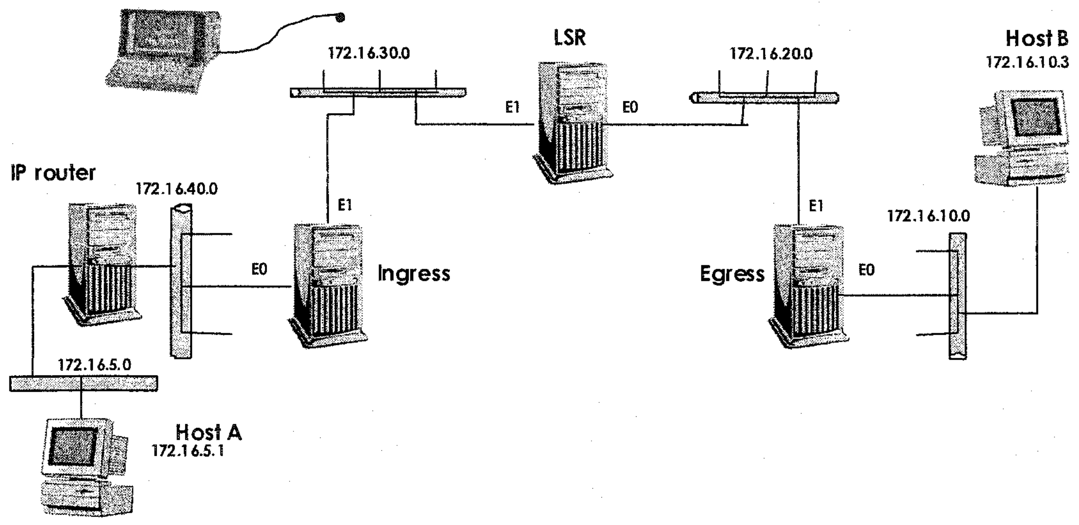

Figure 4-1. Experimental Test-bed set up

In our implementation, RSVP-TE daemon running under each MPLS router is responsible for the RSVP-TE signaling and the maintenance of the MPLS state, e.g. allocation and installation of the MPLS labels during LSP set up and freeing and removing labels on LSP tear down. The Dynamic Class Based Queuing (CBQ) script runs on the core node to dynamically reallocate resources based on the estimation of the future demands since the core router is bottleneck.

4.1.2 MPLS Edge Router Architecture

The Ingress plays a key role in a MPLS Diffserv domain. It is responsible for enforcing the SLA with the customer domain. Figure 4-2 shows the architecture of the dynamic bandwidth allocation functionality. It consists of the control & management plane and the data plane (corresponding to the user space and kernel space of Linux, respectively). The control & management plane includes the following modules:

- Bandwidth Meter Module (BMM): It measures the bandwidth periodically from the traffic and updates the database in real time. It calculates the bandwidth by using the libiptc library function call [103], which is included in the "iptables" package. These functions calls build built-in byte counters in the Linux kernel, which can register each byte that passes. It is very accurate. When we read these byte counters and know the exact time, then we can calculate how many bytes are processed per second (bandwidth). Or we can measure the average sample rate during a certain period.

- Traffic Estimation Module (TEM): It is used to estimate traffic based on on-line measurements. It gets sample data from the BMM In our work we deploy three different traffic estimation algorithms.

- Resource Allocation Module (RAM): It dynamically reallocates the bandwidth based on the estimated value from the TPM. Dynamic CBQ is deployed in our work.

- Database (DB): It contains the sample bandwidth history during the measuring window. Here we use a simple file operation. In order to avoid race condition [104], we use a semaphore mechanism to synchronize between BMM and TEM.

The data plane consists of the Linux traffic control module (*tc*), supported by recent versions of the Linux kernel [63][78]. This module is used to build queuing disciplines, classes and filters. Diffserv classification and traffic conditioning happen here as well [79][80].

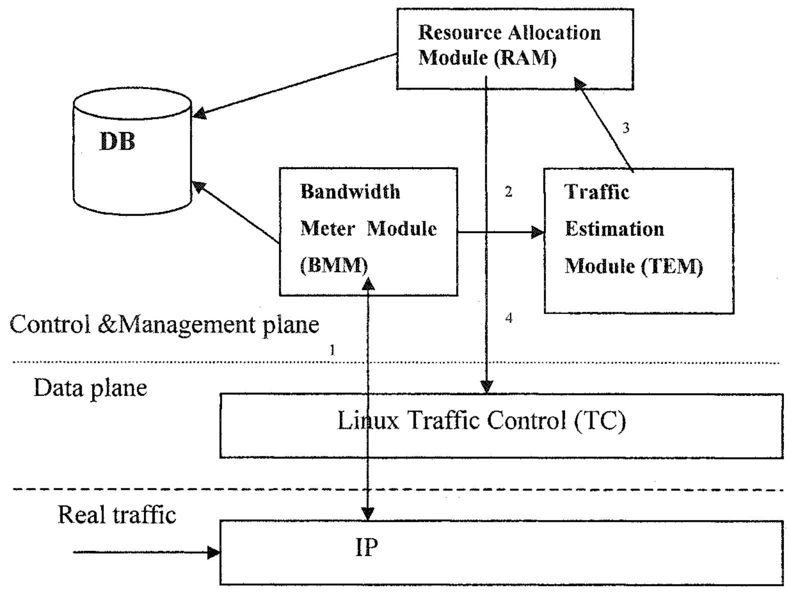
Resource Allocation Module (RAM)
DB
Bandwidth Meter Module (BMM)
Traffic Estimation Module (TEM)
1
2
3
4
Control &Management plane
Data plane
Linux Traffic Control (TC)
Real traffic
IP

Figure 4-2. Edge router Dynamic Bandwidth Allocation modules

4.1.3 Terminology

The frequently used Linux tools in this chapter are defined in Table 4-1.

Table 4-1. Frequently used Linux tool

netfilter/iptables	The "netfilter/iptables" is the Linux 2.4.x / 2.5.x firewall subsystem. It delivers you the functionality of packet filtering, all different kinds of NAT (Network Address Translation) and packet mangling.
tc/iproute2	"tc" is a user-space program used to manipulate individual traffic control elements. Its source is from iproute2 package.
tcindex	A parameter of "tc", used to identify DS field.
ip/iproute2	"ip" is an excellent static and policy based addressing and

	routing configuration tool. Its source is from iproute2 package.
dsmark	A queuing discipline that offers capabilities needed in Diffserv.
cls_tcindex	A classifier to select classes

4.2 LINUX QOS SUPPORT

Low cost, easily scalable multi-Ethernet interfaces and a very impressive variety of QoS tools make Linux a good solution for corporate LANs. For example, an extremely powerful tool "iptables"; an excellent static and policy based addressing and routing configuration tool "ip" and a stable and reliable kernel queue implementation including FIFO, PRIO, TBF [22], SFQ [81], CBQ, WRR, HTB, RED and GRED [80] queues. Linux capacity to configure hierarchical QoS class space implementations makes it very suitable for corporate LANs. Also Linux is the only operating system where you can configure a full compliance differentiated service model implementation.

4.2.1 Traffic Control in Linux

Linux kernels offer a rich set of traffic control functions that make a Linux-based router to be capable of emulating the functionality of most full-fledged commercial routers. We will describe these functions below.

Each NIC (Network Interface Card) on a Linux box is driven by a Network Driver, which controls the hardware. The Network Driver acts as an exchange mechanism of packets between the Linux Networking Code and the physical network The Linux Networking Code can request the Network Driver to send a packet to the physical network (packet leaving the box) or the Network Driver can give a packet it has received from the physical network to the Linux Networking Code (packet entering the box). This approach can be viewed in Figure 4-3. The Linux Networking Code is a building box, which can be refined into more functionalities shown in Figure 4-4 [78][81].

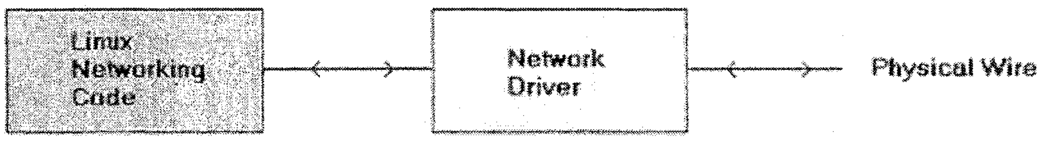

Figure 4-3. Linux traffic control functionality

Figure 4-4 shows roughly how the kernel processes data received from the network, and how it generates new data to be sent on the network. Packets arrive via an input interface, where they may be policed. Policing discards undesirable packets, e.g. if traffic is arriving too fast. After policing, the input de-multiplexer examines the incoming packets to determine if the packets are destined for the local node. If so, they are sent to the higher layer for further processing. If not, it sends the packets to the forwarding block. Those higher layers may also generate data on their own and hand it to the lower layers for tasks like encapsulation, routing and transmission.

The forwarding mechanism includes the selection of the output interface, the selection of the next hop by looking up the routing table and encapsulation etc. Once all this is done, packets are queued on the respective output interface. This is the second point where traffic control comes into play. Traffic control can decide if packets are queued or if they are dropped (e.g. if the queue is out of its length limit, or traffic exceeds some rate limit), also it can decide in which order packets can be sent, it can delay the sending packets etc. After traffic control releases a packet for sending, the network driver picks it up and sends it on the network.

In summary, the Linux traffic control happens in two blocks: Ingress Policing and Output Queuing. Ingress policing is the first point of traffic control that discards undesirable packets to enforce an upper limit on the incoming traffic. The second point will be the Output (Egress) Queuing. Here packets are queued and then dropped, delayed or prioritized according to the policy. The traffic control code in the Linux kernel is just C language code, consisting of the following four major conceptual components:

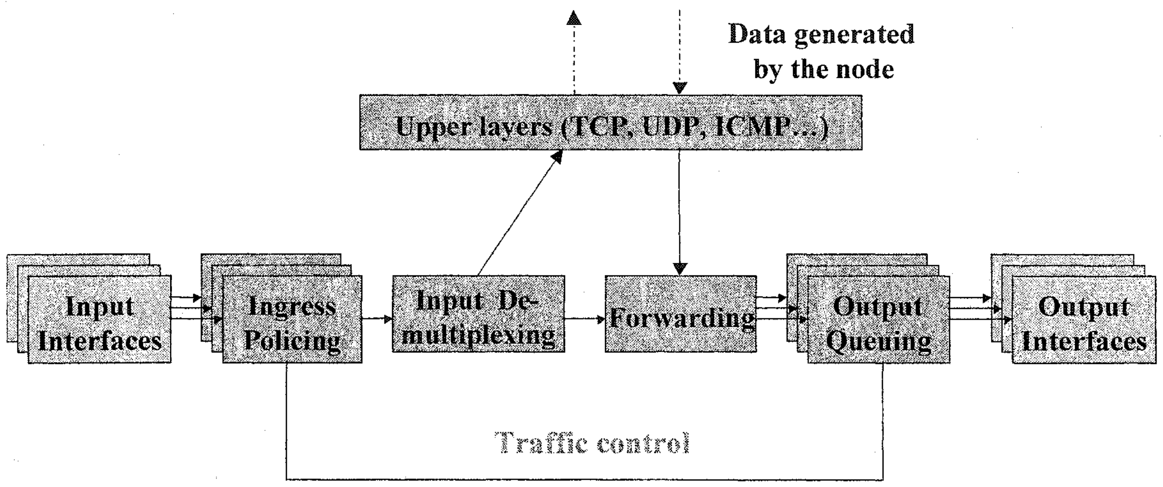

Figure 4-4. Packet processing in the Linux kernel

- Queuing discipline
- Classes (within a queuing discipline)
- Filter
- Policing

Each network device binds a queuing discipline with it, which controls how packets are enqueued on that device. Classes are attached to a queuing discipline and hold a portion of the total traffic. Classes and queuing discipline are linked closely to each other. Typically, classes do not handle packets themselves; rather, have a queue discipline associated with them to perform this task (FIFO is the default queuing discipline).

A more elaborate queuing discipline may use a filter to classify the different classes of packets, processing each class in a specific way. Figure 4-5 is an example of such a queuing discipline. 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. Figure 4-6 shows another example of queuing discipline with two delay priorities (PRIO). Packets selected by the filter go to the high-priority class, while all other packets go to the low-priority class. Whenever there are packets in the high-priority queue, they are sent before packets in the low-priority queue. In order to prevent high-priority traffic from starving low-priority traffic, the Token Bucket Filter (TBF) queuing discipline is attached, which enforces a rate of at most 1 Mbps for example. Finally, the queuing of low-priority packets is done by a FIFO queuing discipline.

Queuing disciplines in Linux can be arbitrarily chosen from a variety of available algorithms (e.g. Priority FIFO, Class based Queuing (CBQ), Token Bucket Filter (TBF), Hierarchical Token Bucket (HTB), Random Early Detection (RED), etc.). Several common queuing disciplines, which are used in our test-bed implementation, have been discussed in Chapter 2.

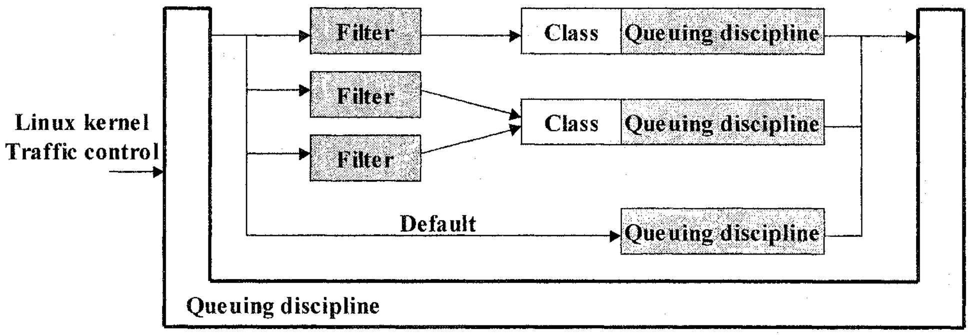

Figure 4-5. Queuing discipline with classes and filters

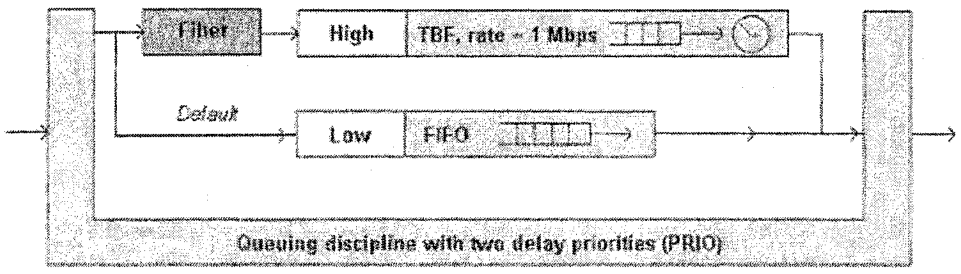

Figure 4-6. Queuing discipline with two delay priorities

4.2.2 Differentiated Services on Linux

Differentiated Services (Diffserv) is an architecture providing different types or levels of service for network traffic. One key characteristic of Diffserv is that flows are aggregated in the network, so that core routers only need to distinguish a comparably small number of aggregated flows, even if those flows contain thousands or millions of individual flows. Support for Diffserv on Linux is part of the more general Traffic Control architecture [79][80].

Support for Diffserv is already integrated into 2.4 kernels. Specifically, three new traffic control elements were added to the kernel in order to support the Diffserv architecture.

- the queuing discipline **sch_dsmark** to extract and to set the DSCP
- the classifier **cls_tcindex** which uses this information to select classes
- the queuing discipline **sch_gred** which supports multiple drop priorities and buffer sharing for Assured Forwarding.

Figure 4-7 illustrates an example of PHB implementation and packet forwarding at a Diffserv node. The BA classifier discriminates packets based on the DSCP value. Those packets marked with 0x2e are forwarded to the class 2:1 with the highest priority. These packets are corresponding to the EF PHB. The remaining unmarked packets are forwarded to class 2:2 with RED queue scheduler.

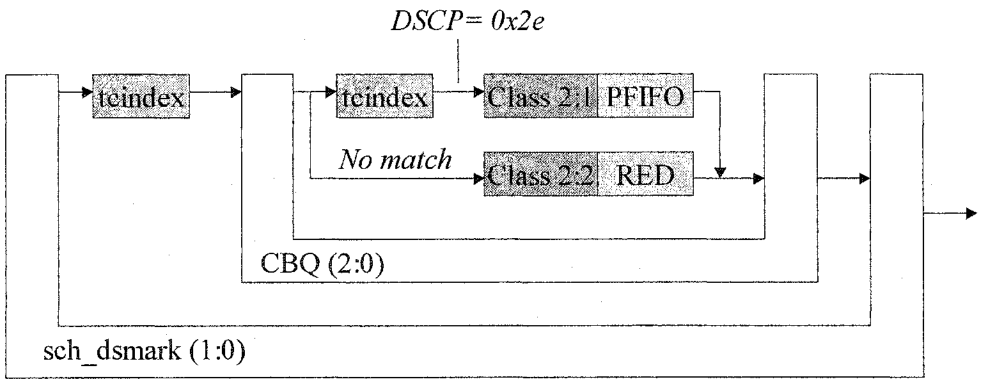

Figure 4-7. Configuration of EF class using CBQ

4.2.3 Diffserv over MPLS under Linux

"RSVP-TE daemon for Diffserv over MPLS under Linux" is an open source project developed by Gent University [55]. As we said before, our test-bed setup is based on software developed in this project. The overall architecture of this project can be found in

[105]. It served as the baseline in the pursuit for end-to-end QoS. In our implementation, the RSVP-TE daemon is responsible for the RSVP-TE signaling and the maintenance of the MPLS state, such as allocation and installation of the MPLS labels during LSP set up and for freeing and removing labels on LSP tear down. We setup an IP router, which runs "iptables" to mark incoming traffic before the traffic enters the MPLS network (see Figure 4-1). The LSP in our MPLS network is pre-configured by using "rtest2" and "rapirecv_auto" which are two components using the RSVP API (RAPI) to build user application interacting with the RSVP daemon. The Dynamic Class Based Queuing (CBQ) script runs on the core node to dynamically reallocate resources based on the estimation of future demands.

4.2.3.1 Packet Forwarding Path

In the Ingress router the packets are classified with netfilter [101][102]. Packets are filtered on the OUTPUT or PREROUTING chain of the mangle table. The mangle table is needed because the fwmark (firewall marks) needs to be set. The OUTPUT and PREROUTING chains are used in order to filter both locally generated and incoming traffic. Based on the value of the fwmark a routing table is selected (using policy routing [106]). In the resulting routing table there is a MPLS tunnel interface acting as the default gateway. The tunnel interface encapsulates the packets on the LSP by attaching the correct outgoing label. The incoming DSCP is mapped to the EXP field in the MPLS header.

In the core node, the MPLS stack inspects the incoming label and sets the new outgoing label and next hop. At the Diffserv level the current EXP value of the packet is inspected. There is a mapping from this EXP value to a tcindex. The tcindex in turn determines the correct outgoing queue so that the correct PHB can be applied.

Finally in the egress the incoming EXP field is mapped to the DSCP field and then the MPLS header is stripped off and the packet is sent to the IP layer.

As we described in Chapter 2, the E-LSP can support up to eight classes. Table 4-2 is an example to show how E-LSP works in our Linux test-bed. The Ingress router looks at the incoming packet's DSCP to identify the packet's BA, and then picks the LSP/Label that supports the right FEC and the right BA. Finally, it marks the EXP field to reflect the packet's BA. The Core router will implement PHB forwarding based on EXP. Under Linux, tcindex is used to identify the packet's PHB at the Diffserv level. So there exists a mapping from EXP field to tcindex at the Core router. In the egress router, the EXP is mapped back to DSCP.

Table 4-2. E-LSP strategy implemented in our Linux environment

Ingress		Core		Egress	
DSCP (6 bits)	EXP	EXP	tcindex (8 bits)	EXP	DSCP
(BE) 0x0	0	0	0xffff	0	0x0
(EF) 0x2e	1	1	0xb8	1	0x2e
(AF11) 0x0a	2	2	0x28	2	0x0a
(AF12) 0x0c	3	3	0x30	3	0x0c
(AF21) 0x12	4	4	0x48	4	0x12
(AF22) 0x14	5	5	0x50	5	0x14
(AF31) 0x1a	6	6	0x68	6	0x1a
(AF32) 0x1c	7	7	0x70	7	0x1c

4.3 NETWORK PERFORMANCE MEASUREMENT TOOLS

To carry out accurate measurements on QoS parameters in our local network, we employed a number of performance measurement tools. These include hardware-based InterWatch95000 (IW95000) network protocol analyzers [82] and some software-based packet-capture tools, such as Ethereal/Tethereal [75] and Tcpdump [76].

4.3.1 GN Nettest's IW95000s

GN Nettest's InterWatch 95000 (IW95000) 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 IW95000 can be connected simultaneously to multiple Ethernet segments, while monitoring network performance (e.g. throughput and latency).

In our experimental work, two network testing units have been used. One acts as background traffic generator, the other is used to monitor the traffic.

4.3.2 Network Sniffer/Analyser

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

Tethereal (terminal version of ethereal) is used to capture MPLS packet data from a live network. Based on the log data, we can parse and analyze the network performance, for example, delay, jitter and packet loss rate etc. The traditional Tcpdump application is used to measure the IP network performance.

4.4 TRAFFIC SOURCES

Modeling and analysis of computer network traffic has been an area of extensive research over the past several years, especially with the emergence of new Internet applications with varying resource and QoS requirements [91][92][93][94].

The use of realistic traffic 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:

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. A lot of research has been focused on obtaining realistic models for the traffic generated by the users of telecommunications networks. Self-similarity and Long-Range Dependence (LRD) are the key properties of aggregated traffic, which have different statistical properties than Short-Range Dependence (SRD) processes. Several models of this type have been proposed [91][92][93][94][96][97]. However, these models are unable to capture the high level of burstiness of some types of traffic flows. The authors of [60][95] proved that α–stable long-range dependent stochastic processes more accurately describe aggregate traffic passing through modern networks as compared to earlier models, especially for aggregate traffic that presents very high burstiness. The classification of a stochastic process into those with short or long-range dependence is based on the asymptotic properties of the autocorrelation function [94].

In our experimental tests, we adopted two types of artificial traffic sources: α–stable LRD traffic generator [60][95] and SRD UDP Poisson traffic [92][96].

4.4.1 α–stable Long Range Dependent Traffic

The main purpose of our α–stable LRD traffic generator is to regenerate the α–stable LRD traffic in our Linux test-bed to see how the system behaves under this type of traffic.

There are three important parameters to model α–stable LRD traffic: type of the stochastic process, α and H [95]. The Fractional Stable Noise (FSN) process is adopted because it can model high variability and long-range dependence at the same time, something that the Fractional Gaussian Noise (FGN) model fails to do (it lacks the ability to capture high variability). Three modes of FSN are defined: balanced Linear Fractional Stable Noise (LFSN), anti-balanced LFSN and Log-FSN. Each model allows reproduce more accurately traffic produced by certain applications (e.g. video, ftp, web etc.). α (alpha) is called the index of stability, its value affecting the level of burstiness $\alpha \in (0,2]$. When α=2, we revert to a Gaussian process. Smaller α produces higher burstiness. H is called Hurst parameter or self-similarity parameter. It is positive and upper bounded by a value that depends on the marginal distribution of the process ($H \leq 1$). Higher H, produces a strong autocorrelation. $H = 1/2$ for SRD, $1/2 < H < 1$ for LRD.

The α–stable traffic generator is developed based on two traffic models proposed in [60]. Following [60], each traffic process is modeled as a function of a fractional stable noise process Y_j. For instance, the following model has been proposed for aggregate MPEG video traffic:

$$W_j \overset{d}{=} m + aY_j \;\; ; \;\; j \in \{0, 1, 2, \dots\} \qquad (4.1)$$

where W_j represents the number of arrivals during the j-th time interval of unit length. Y_j is FSN with index of stability α and Hurst parameter H. The parameter m is the mean of the number of arrivals per unit time and a is a scaling factor. A slightly modified version of the previous model was proposed in the same work [60] for Ethernet and WWW traffic:

$$W_j \overset{d}{=} m + a(| Y_j | - \mu_Y) \;\; ; \;\; j \in \{0, 1, 2, \dots\} \qquad (4.2)$$

where μ_Y is the mean value of $|Y_j|$. The model components m, a and Y_j have the same meaning as in Equation (4.1). The values for all the model parameters have to be estimated off-line from actual traffic samples.

Five types of traffic can be generated based on these two models. The model parameter is shown following [60][95]:

- Highly bursty WAN traffic (Log-FSN, α = 1.05, H =0.997)
- Moderately bursty Ethernet traffic (balanced LFSN, α = 1.95, H =0.834)
- WWW traffic (anti-balanced LFSN, α = 1.28, H =0.833)
- MPEG1 VBR traffic (balanced LFSN, α = 1.95, H =0.903)
- MPEG2 VBR traffic (anti-balanced LFSN, α = 1.93, H =0.824)

We wrote a C language code to generate α-stable LRD traffic using α–stable synthetic traffic trace files. The functionality of the code can be broken down to two parts: i) generating α–stable synthetic trace file and ii) sending the traffic over the network by using

UDP socket. The first part is done based on the appropriate traffic model and the use of the proper parameters.

The program initially asks the user to enter the various parameters like the destination address, destination port number and average rate, then ask the user to choose the type of traffic for each connection. For example, when you choose the simulated MPEG-1 traffic, the corresponding FSN trace file with parameters (balanced LFSN, $\alpha = 1.95$, $H = 0.903$) will be chosen, and the artificial trace file is created based on the traffic model. The program will then send UDP traffic based on the artificial trace file.

Figure 4-8 and 4-9 shows the traces resembling α–stable traffic with $\alpha = 1.95$ and $\alpha = 1.60$. Observe that α–stable traffic with $\alpha = 1.60$ demonstrates high variability statistics. In the performance analysis chapter, we apply these two kinds of traffic generated by our α–stable generator. In order to achieve the statistical accuracy for α–stable distributions, one FSN trace is not enough. We generate several traces with different "seeds" but the same model parameters. For performance evaluation, we collect the results with different traces and take the average.

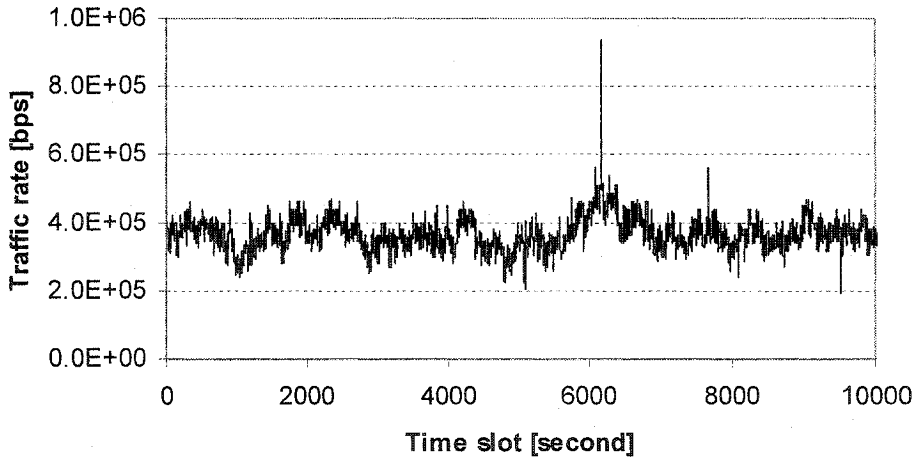

Figure 4-8. Trace of α–stable traffic with $\alpha = 1.95$

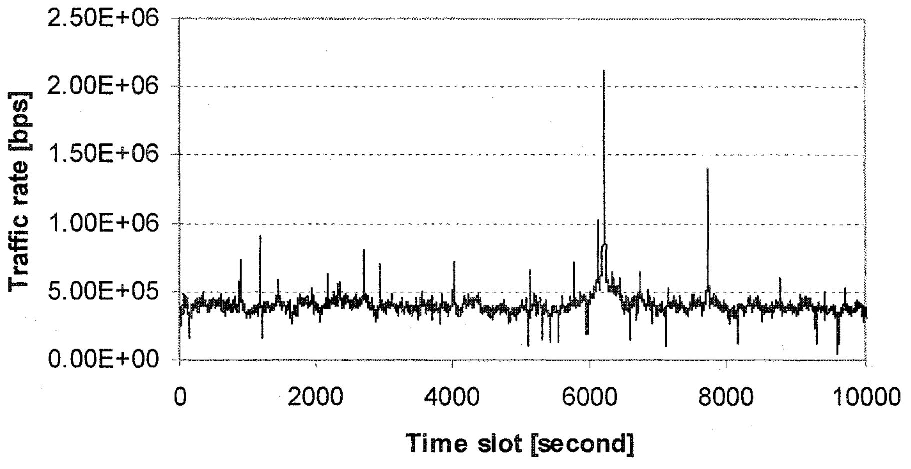

Figure 4-9. Trace of α–stable traffic with α = 1.60

Also in order to confirm the statistical properties of the traffic stream generated by our α–stable LRD traffic generator, we record the traffic stream and analyze these traces. For detailed validation of α–stable LRD traffic source refer to Appendix A2.

4.4.2 Short Range Dependent UDP Poisson Traffic

The Multi-Generator (MGEN) [83] is chosen to generate the short-range dependent Poisson traffic in our experiments. The MGEN is an open source software by the Naval Research Laboratory (NRL) PROTocol Engineering Advanced Networking (PROTEAN) Research Group. MGEN provides the ability to perform IP network performance tests and measurements using UDP/IP traffic.

The MGEN can generate real-time traffic patterns so that the network can be loaded in a variety of ways, such as Poisson and Periodic traffic. The generated traffic can also be received and logged for analysis. Script files are used to drive the generated loading patterns over the course of time. These script files can be used to emulate the traffic patterns of unicast and/or multicast UDP/IP applications. Currently MGEN runs on various Unix-based (including MacOS) and WIN32 platforms. Figure 4-10 shows an example of Poisson traffic pattern with mean rate 0.8Mbps generated by MGEN.

Again for the detailed validation of Poisson traffic source please refer to Appendix A1.

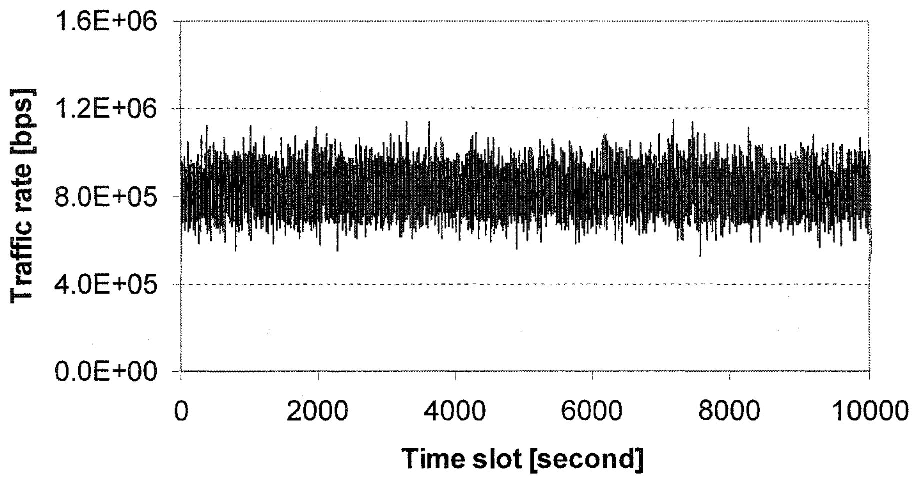

Figure 4-10. Trace of UDP Poisson traffic produced with MGEN

Chapter 5

MEASUREMENTS AND PERFORMANCE ANALYSIS

This chapter provides thorough performance evaluation of our dynamic bandwidth allocation scheme. Section 5.1 gives the overview of our work. Section 5.2 evaluates the Maximum Estimator (**ME**). The Gaussion Estimator (**GE**) is evaluated in Section 5.3. Section 5.4 evaluates the non-Gaussian α–stable Estimator (**ASE**). Finally Section 5.5 concludes our work.

5.1 INTRODUCTION

We have run extensive experiments to explore the effectiveness of our estimators. For each performance value we repeated each experiment 5 times in order to capture the traffic statistics more accurately. All results shown here are based on the assumption that we only have one VPN tunnel and one traffic class. The network topology is shown in Figure 4-1. In this set of experiments we send traffic from Host A to Host B through one Diffserv-enabled LSP. We estimate the traffic at the Ingress router, and then change dynamically the CBQ rate at the Core router. The Core router is the bottleneck.

The estimators evaluated in this work are the Maximum Estimator (**ME**), the Gaussion Estimator (**GE**) and the non-Gaussian α–stable Estimator (**ASE**). The algorithms are described in Chapter 3. The traffic sources deployed here are UDP Poisson traffic (generated by MGEN [83]) and α–stable traffic [60] (generated by our α–stable traffic generator). We chose these two types of traffic in order to evaluate the schemes under two extreme conditions: short-range dependent (memoryless) traffic, and long-range dependent, highly bursty traffic. Because of processing limitations of the Host PC, used to generate the traffic, we keep the generated traffic volumes to low values: 0.8 Mbps for the Poisson traffic and 0.4 Mbps for α–stable traffic. Higher transmission rates result in deviations from the expected statistical behavior. More detail on this and the validation of the traffic sources can be found in Appendices A1 and A2.

The QoS parameters we measure are: packet loss rate, average delay, delay jitter and the ratio of average reserved bandwidth over the average traffic volume. Since our backbone network in this case consists of only three routers, all traffic goes through the same path. The bottleneck resides within the Core router. All other routers are operating at fast Ethernet, thus as long as the routing related processing does not become a bottleneck due to the processing limitations of the PC, there should not be affecting the traffic characteristics and the delays they introduce should be small. According to the results we have shown in Appendix B, the ingress router operating in the MPLS Diffserv mode has problems when the traffic exceeds 5 or 6 Mbps, thus, we ensure that the peak traffic remains bellow this threshold. We measure the average forwarding delay, delay jitter, and packet loss, within the core router.

The system is evaluated for three traffic sources:

- Poisson traffic (short-range dependence)
- α-stable traffic with $\alpha = 1.95$ (long-range dependence)
- α-stable traffic with $\alpha = 1.60$ (long-range dependence)

Please note that in our experiments, the average amount of produced traffic does not change, as will the case be in a real environment. The objective of the system described and analyzed in this dissertation is to capture these changes of resource requirements and adjust the reservation accordingly. For this purpose, the traffic estimators have to be used, which process traffic measurements over a certain period T_{meas} and apply the reservation over a period Y. It is quite clear that should a change occur, it would be beneficial to track it as soon as possible. This requires that T_{meas} and Y are small. However, the randomness of the traffic tends to produce erroneous measurements for short observation window. In addition, should the updating of reservation policies (controlled by the value of Y) are infrequent, the system will not be able to capture (and benefit from) changes in traffic needs fast enough. Should the traffic needs become reduced, resource will be wasted, since they will be maintained reserved, yet not used. On the contrary, if the traffic volume increases, should the adjustment in bandwidth does not occur fast enough, massive packets losses will be occurring, seriously impairing (and possibly terminating) the real time applications. Thus, it is important that the value of the performance related parameters are chosen properly. In our analysis, we attempt to identify the values that allow us to make reasonably decisions (and apply successful policies) at the shortest period possible, when the randomness of the traffic is taken into consideration.

The packet loss and forwarding delay are associated with the amount of buffer allocated within the routers and switches. Buffering offers in a sense traffic smoothing. Thus, the buffer size B should be investigated as well and access its impact on performance. In summary, the parameters we examine in terms of their impact on the performance:

- Buffer size at the router(s) B
- Sample interval X
- Measurement window T_{meas}
- Resource reservation window Y

For each parameter, we plot:

- The amount of packet losses[1] at the Core router
- The average forwarding delay at the Core router

[1] ———————————

[1] Please note, in order to illustrate the difference more clearly especially for the small value of packet loss, we provide two curves regarding packet loss: one is namely the numerical value, the other is the logarithm scale value.

- The average delay jitter (IPDV[2]) at the Core router
- The ratio of average reservation over the average traffic volume

Class Based Queuing (CBQ) is applied to perform resource reservation. In order to accurately measure these QoS metrics, we utilize an external analyzer to monitor in promiscuous mode the incoming traffic in the subnet of the Core router, and record the arrival-time of each incoming packet. At the same time, the analyzer performs the same action at a second interface on the outgoing subnet of Core router for packets departing the Core router. This approach avoids the clock synchronization problem using same clock to monitor arrival and departure time of each packets. However, this method is only suitable for a laboratory environment, in real-life scenario the time synchronization is needed.

5.2 EVALUATION OF MAXIMUM ESTIMATOR (ME)

Let us recall that ME operates as follows:

$$R_{ren} = \max\{R_i\}$$

To further enhance the generality of the system, we introduced the following modification:

$$R_{ren} = \max\{R_i\} * (1 + M)$$

where R_{ren} is the renegotiated rate for the next reservation interval, R_i is the average rate over the inter-sample interval and M is the margin. The margin values used in the experiments are: 0, 5% and 10%. We conduct the following sets of experiments. Please note that the results for both Poisson and α–stable traffic are presented.

Experiment 5-1

[1]

[2] Instantaneous Packet Delay Variation (IPDV), formally defined by the IP Performance Metric (IPPM) working group, is based on the one-way delay measurements. The measurement of a single IPDV is defined as the difference of delays experienced by consecutive packets. $IPDV_{jitter} = | D_t - D_{t-1}|$

The objective of this set of experiments is to explore how different values of the sample interval X affect the system's performance. For this purpose, we choose $Y = 1$ min and $T_{meas} = 5$ min. We apply a buffer size of 10 packets and 100 packets at the core router. We run experiments using the margin value $M = 0$ for X =: 5 seconds, 10 seconds, 15 seconds and 20 seconds and plot the performance curves versus X.

The value of X controls the sample interval. In order to capture the variability of a real traffic process, it is better to choose small X, i.e. 1 second. However, smaller X means the number of samples is larger when T_{meas} is constant. Taking into consideration the processing load, the calculations will generate to the router, we decided to use for the experiments value of X starting from 5 seconds. The ME algorithm picks the maximum sample value during the T_{meas} as the new allocation rate for the next time period Y. Since use of larger X tends to produce an average and reduce the traffic spikes, the allocation rate tends to be smaller as the value of X increases. Figure 5-1 and Figure 5-5 show the ratio of reservation over average traffic volume for Poisson and α-stable traffic, which confirm that the allocation rate decreases when increasing the value of X. Observe that the ratio for α-stable traffic is higher as compared to Poisson traffic because of the higher burst nature of α-stable traffic. However, as the reservation decreases with the increase of the X value, the performance will either stay at the same level or deteriorate.

Figure 5-2 and Figure 5-6 show the average forwarding delay for Poisson and α-stable traffic. The results for a buffer $B = 10$ packets indicate lower average delay for α-stable traffic as compared to Poisson, yet, the results for a buffer $B = 100$ packets indicate the opposite. The explanation is that by nature, α-stable traffic generates most of the bursts to be intensive when the buffer is small. This results in fast filling of the buffer with the obvious consequence of experiencing massive losses. These losses can be verified at the loss rate curve, but lost packets do not contribute to the delay. In a sense, we are getting situations where the buffer is either empty or almost empty, or filled and packets become lost. When the buffer increases, these long intensive bursts are absorbed in the buffer, and the losses reduce. However, these longer fills of buffer generate higher average delay.

Figure 5-3 and Figure 5-7 present the IPDV jitter for Poisson and α-stable traffic. The jitter for the case of Poisson is higher as compared to α-stable traffic. The possible reason is the memoryless nature of the Poisson traffic and fact that each period is statistically independent from the other. Thus, the delays experienced by close-by packets can be significantly different, thus generating high IPDV. For α-stable traffic, the bursts are consistent, thus packets experience similar delays and the difference remains close to zero, giving smaller IPDV.

Figure 5-4 and Figure 5-8 show the packet loss rate for Poisson and α-stable traffic. For buffer B =10 packets, Poisson shows higher losses than α-stable traffic. This might be due to the fact that we are using 0.8 Mbps rate for Poisson, whereas the α-stable traffic is only 0.4 Mbps, thus the short buffer fills for Poisson and produced losses. However, the Poisson bursts are mostly short and when the buffer increases, almost all of them are absorbed. However, α-stable traffic, by its nature, can produce very long bursts with finite probability. When these occur, the buffer fills and losses occur, even at buffer size of 100 packets. These results confirm that α-stable traffic is a serious challenge for networks.

In summary, smaller X will achieve better system performance, especially for the α-stable case. We set X = 5 seconds for following set of experiments.

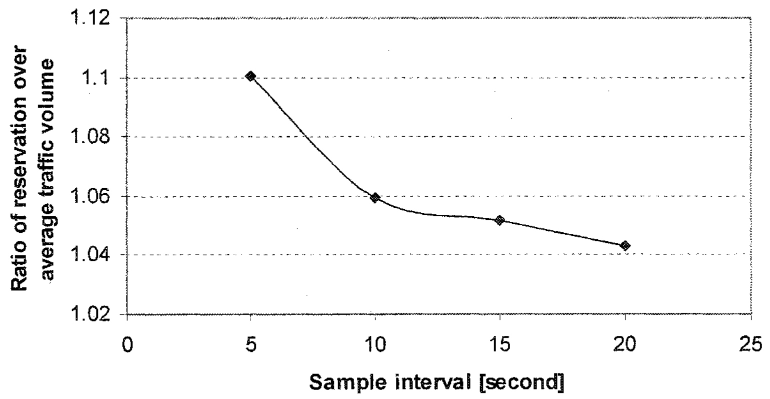

Figure 5-1. Ratio of reservation over average traffic volume vs. X when Poisson traffic is applied and the ME is used

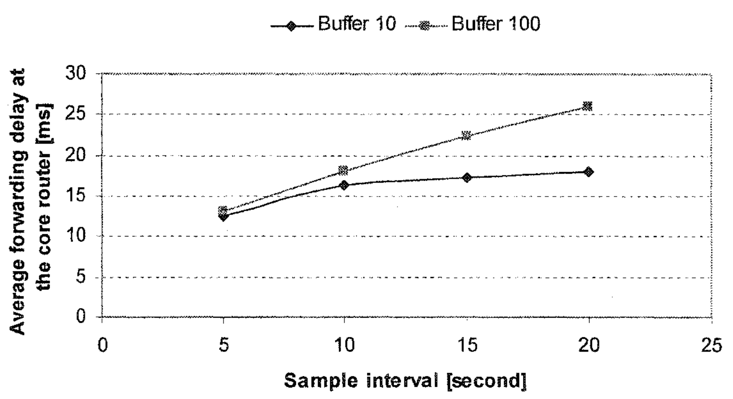

Figure 5-2. Average forwarding delay [ms] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and ME is used

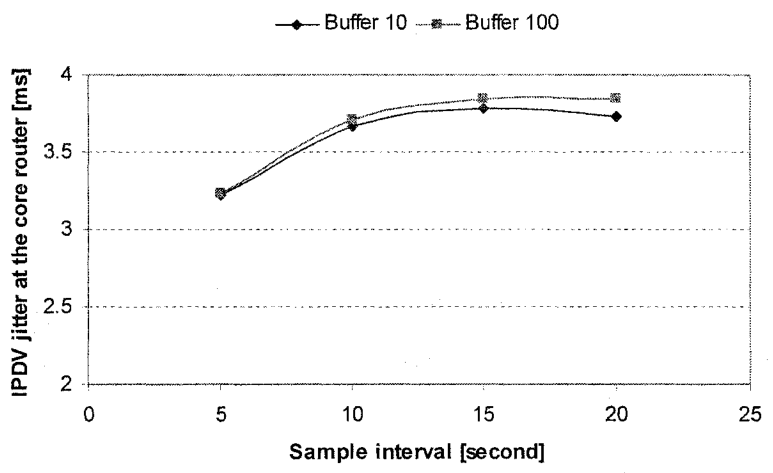

Figure 5-3. IPDV jitter [ms] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and the ME is used

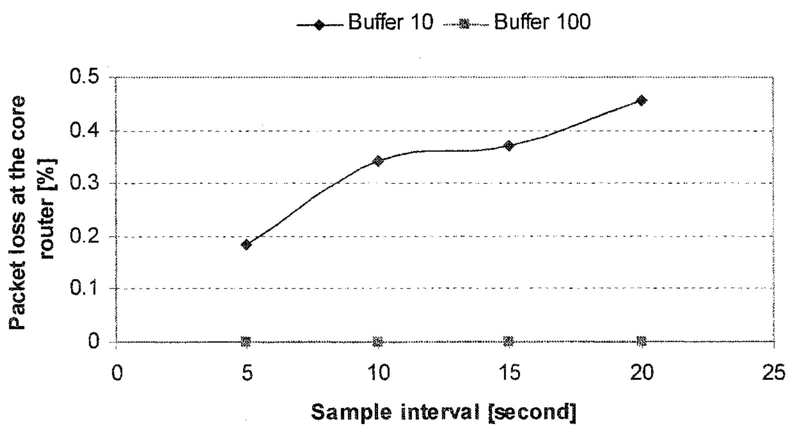

Figure 5-4a. The numerical value of packet loss [%] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and the ME is used

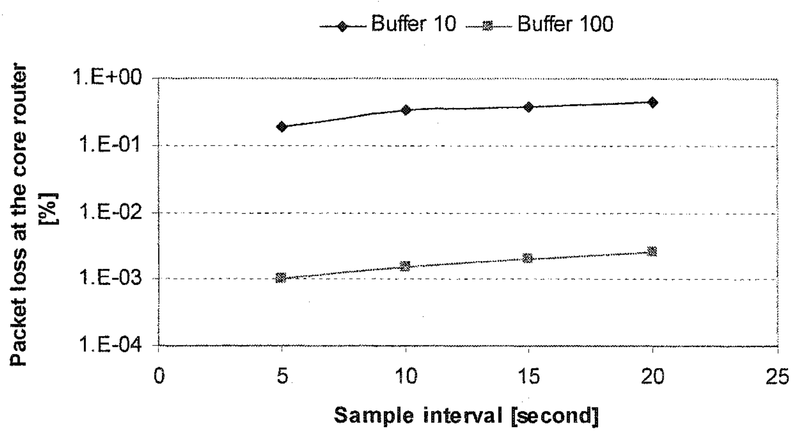

Figure 5-4b. The logarithm value of packet loss [%] measured at the core router vs. X with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and the ME is used

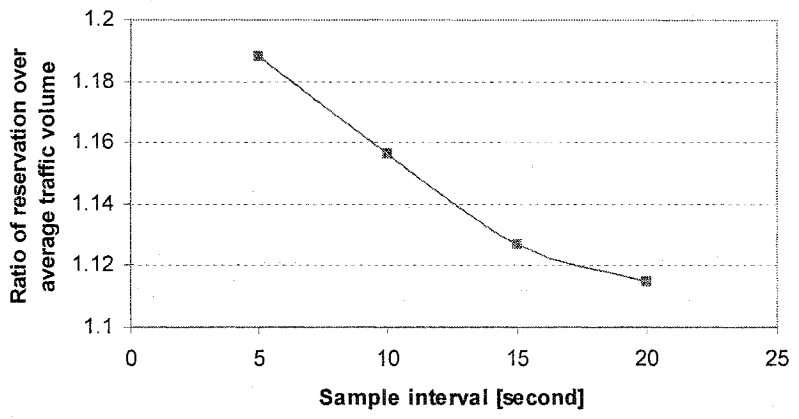

Figure 5-5. Ratio of reservation over average traffic volume vs. X when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

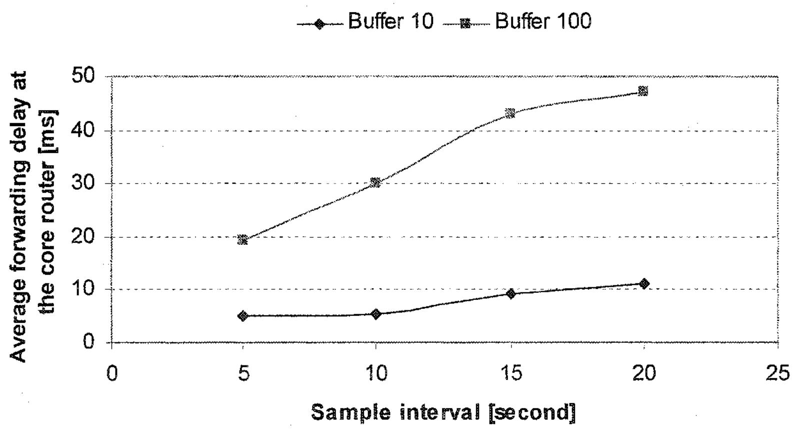

Figure 5-6. Average forwarding delay [ms] measured at the core router vs. X with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

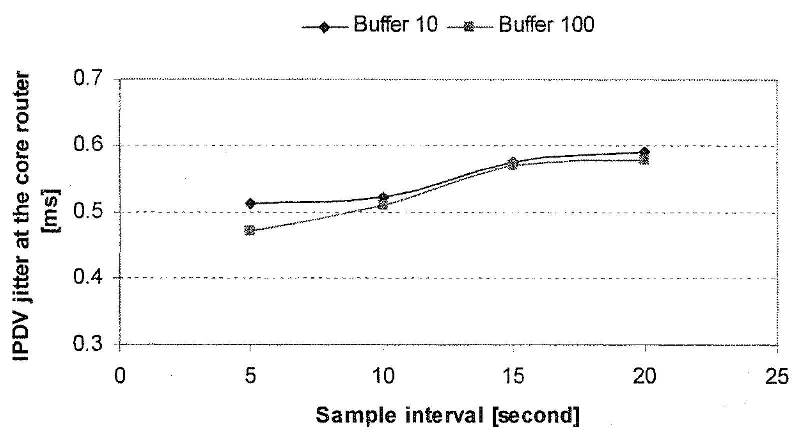

Figure 5-7. IPDV jitter [ms] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

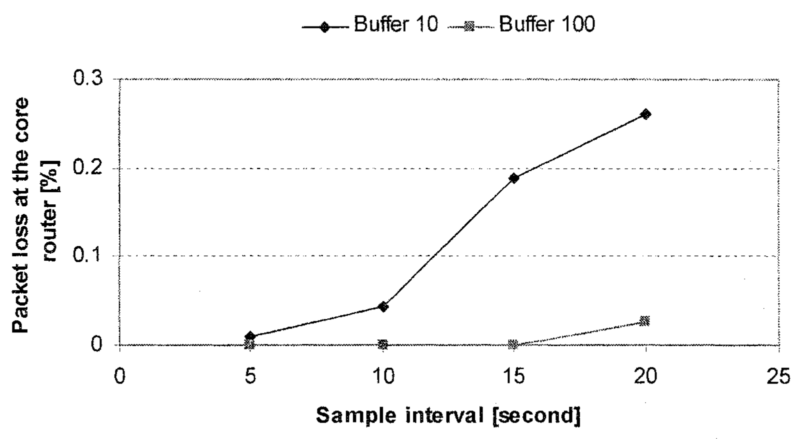

Figure 5-8a. The numerical value of packet losses [%] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

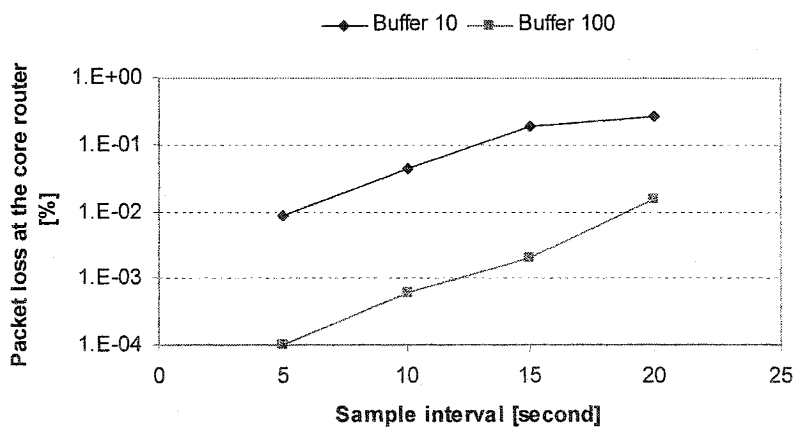

Figure 5-8b. The logarithm value of packet losses [%] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

Experiment 5-2

The objective of this set of experiments is to assess the effect the size of the measurement window T_{meas} and margin have on the performance. We set as $X = 5$ seconds, buffer size $B = 10$ packets and $Y = 1$ min. We conduct experiments for $T_{meas} = 1$ min, 2 min, 3 min and 5 min (number of samples W = 12, 24, 36 and 60). The margin size M we choose is 0, 5% and 10%. Here we evaluate three kinds of traffic patterns: Poisson, α–stable traffic with $\alpha = 1.95$ and α–stable traffic with $\alpha = 1.60$.

The value of T_{meas} controls the number of samples within the measurement window. Larger T_{meas} provides more samples and also provides higher chance to get a higher value of allocation rate for the next time period. Since ME picks the highest value within the observation window, the ratio of reservation over average traffic volume increases when T_{meas} increases. Larger ratio will result in better system performance, the average forwarding delay, IPDV jitter and packet loss decreases.

Figures 5-9, 5-13 and 5-17 show the ratio of reservation over average traffic volume for Poisson and α–stable traffic with $\alpha = 1.95$ and $\alpha = 1.60$. Observe that the ratio value of α–stable traffic is higher than the Poisson, and for $\alpha = 1.60$ it becomes the highest. The explanation is as follows. Since α–stable traffic is burstier than Poisson, it tends to produce higher values of samples. The burstiness of the $\alpha = 1.60$ traffic is even higher as compared to $\alpha = 1.95$, thus the effect of burstiness will be further exemplified. Thus $\alpha = 1.60$ traffic demonstrates a tendency of producing higher levels of reservation. For example, when $M = 0$, and $T_{meas} = 1$ min, the ratio values for Poisson and α–stable traffic with $\alpha = 1.95$ and $\alpha = 1.60$ are 1.075, 1.096 and 1.106 respectively.

Figures 5-10, 5-14 and 5-18 show the average forwarding delay for Poisson and α–stable traffic with $\alpha = 1.95$ and $\alpha = 1.60$. It is evident that an increase in T_{meas} reduces the delay more in α–stable traffic as compared to Poisson. For α–stable traffic, longer window is more beneficial. However, for Poisson traffic, it seems that 2 or 3 minutes window is enough. For example, for $M = 0$ and T_{meas}=1 min, all cases demonstrate similar levels of delay (around 16 ms). However, when T_{meas} changes from 1 min to 5 min, the delay for Poisson drops down to 13 ms, the delay for $\alpha = 1.95$ drops down to 5 ms, and the delay for $\alpha = 1.60$ drops to 4 ms. The explanation is as follows. The statistical information gathered over a wider window for Poisson does not provide any additional accurate knowledge, due to the fact that Poisson is a memoryless process. If higher reservation occurs due to some spikes occurring within the window, does not mean that high spike will appear within the following resource allocation period. This means that when having Poisson traffic, higher allocation, in many cases is of no use, and the extra resource is wasted. Also, if there is low resource allocation within a period, there is a strong chance there will not be need for extra resource in the next period. On the contrary, α–stable process has memory. This means that if higher burst is detected, then there is a good chance the following time period will be experiencing high burstiness as well, thus resource is needed. If on the contrary, the results indicate low burstiness, it is again quite possible the following period will not be experiencing high burstiness, thus the reduction of reserved resource will not penalize the performance. Of course, it is evident that we need a wide window to be able to track this tendency in performance. 1 minute seems to be small. The results indicate that increasing the window size T_{meas} provides substantial improvement to the α–stable traffic system.

A very interesting observation comes from the comparison of the M = 5% and M = 10% curves of Poisson and α–stable traffic. It is evident that α–stable traffic takes more benefit from the higher margin. For M =5 %, when T_{meas}= 1 min, the average delay of Poisson is 13 ms, whereas it is only 7 ms for $\alpha = 1.95$ and 5.5 ms for $\alpha = 1.60$. As far as comparing $\alpha = 1.95$ with $\alpha = 1.60$, the average delay experienced by the $\alpha = 1.60$ traffic reduces faster than that experienced by the $\alpha = 1.95$ traffic with the increase of T_{meas}. This happens because of the fact that $\alpha = 1.60$ exhibits higher burst period, which produces more losses, however, these lost packets are not counted in the delays. Packets arrive in "clusters" of higher bursts, and get dropped. There are also longer inactivity periods (low traffic) as compared to the $\alpha = 1.95$ case. During these periods, the delays are very small.

Figures 5-11, 5-15 and 5-19 show the IPDV for Poisson and α–stable traffic with $\alpha = 1.95$ and $\alpha = 1.60$. Observe that IPDV for Poisson is higher as compared to the α–stable case. Also, higher margin provides higher benefit to the α–stable than Poisson. Again, the reason could be the memoryless nature of the Poisson traffic and the fact that each period is independent from the other. The delays experienced by close-by packets can be significantly different for Poisson traffic, generating high IPDV. For α-stable traffic, the bursts are consistent. Packets experience similar delays and the difference remains close to zero, giving smaller IPDV.

Figures 5-12, 5-16 and 5-20 show the packet loss rate for Poisson and α–stable traffic with $\alpha = 1.95$ and $\alpha = 1.60$. It is clear that Poisson is suffering the most, $\alpha = 1.60$ less and $\alpha = 1.95$ the least. It seems that the memoryless nature of Poisson does not help much when applying dynamic reservation. On the contrary, α–stable traffic benefits by the dynamic reservations because of the memory existing within the statistical behavior of the traffic. The $\alpha = 1.60$ traffic has higher packet loss rate than $\alpha = 1.95$, for the same with that presented in the average delay analysis.

Based on the results and produced analysis, we draw the following conclusions:

- It appears that the estimator is of more help in the case of α–stable than Poisson traffic. The reason is that in addition to be able to track "long term" changes, it can also in a sense detect the presence of long bursts and request the resource needed.

- It appears that it is more beneficial to a α–stable process with lower degree of burst. This is due to the fact that the lower α generates longer and more intense periods of high activity and low activity. Significant losses will be occurring during the beginning of such high burst periods, because the estimator has not captured them yet, and the reservation has not changed.

- It appears that 2 to 3 minutes T_{meas} window is enough for Poisson. However, for α–stable traffic, longer window T_{meas} seems to help. It takes a longer window to reach the steady state level (lower level of losses).

- Use of higher margin seems to benefit the α–stable process more.

In summary, larger value of T_{meas} and higher margin provides better performance, especially for α–stable traffic.

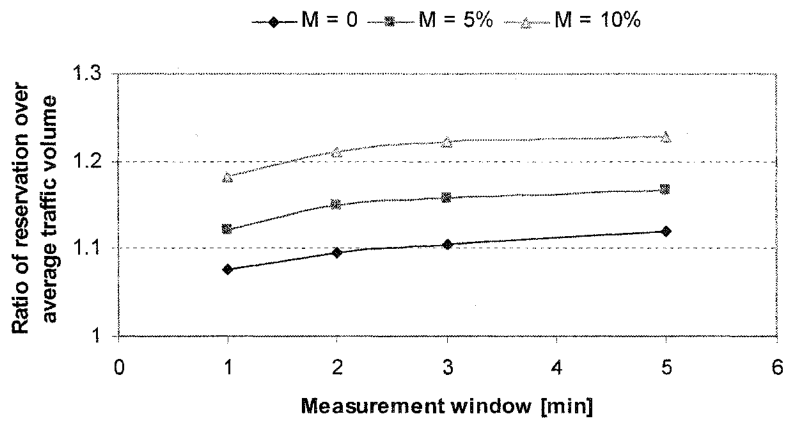

Figure 5-9. Ratio of reservation over average traffic volume vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the ME is used

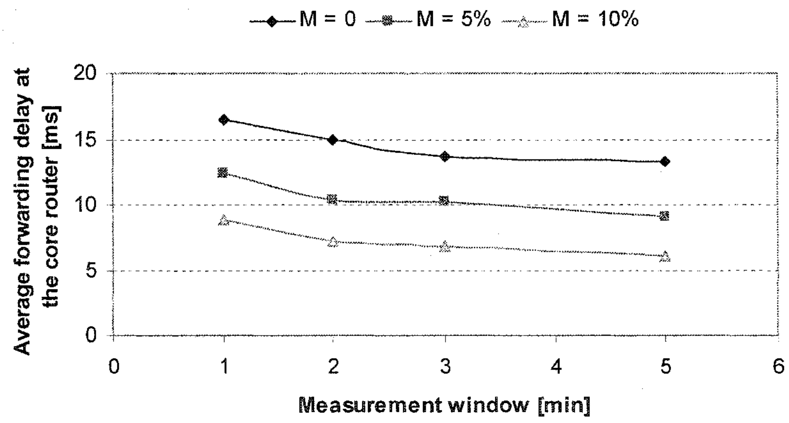

Figure 5-10. Average forwarding delay [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the ME is used

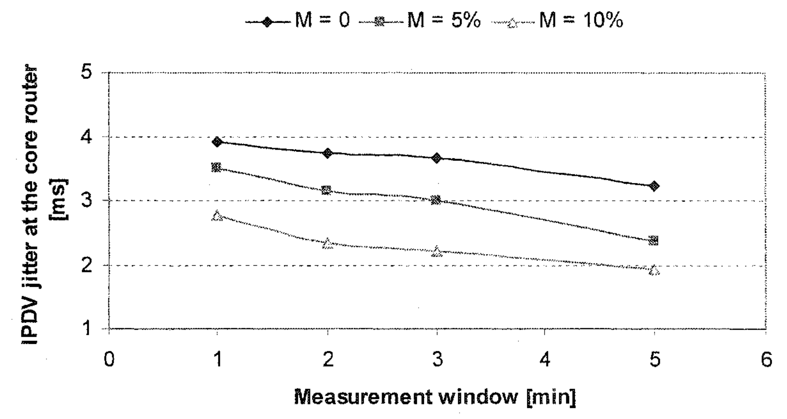

Figure 5-11. IPDV jitter [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the ME is used

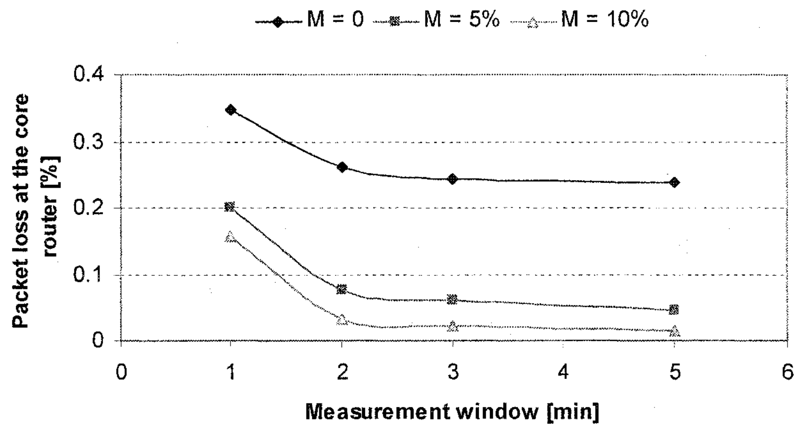

Figure 5-12a. The numerical value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the ME is used

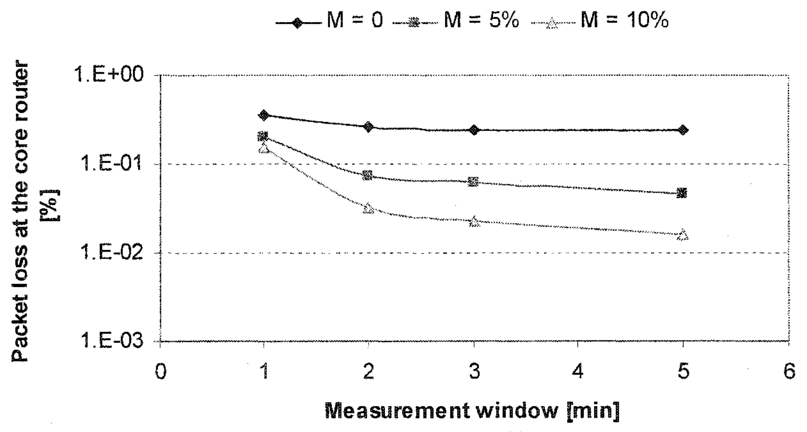

Figure 5-12b. The logarithm value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the ME is used

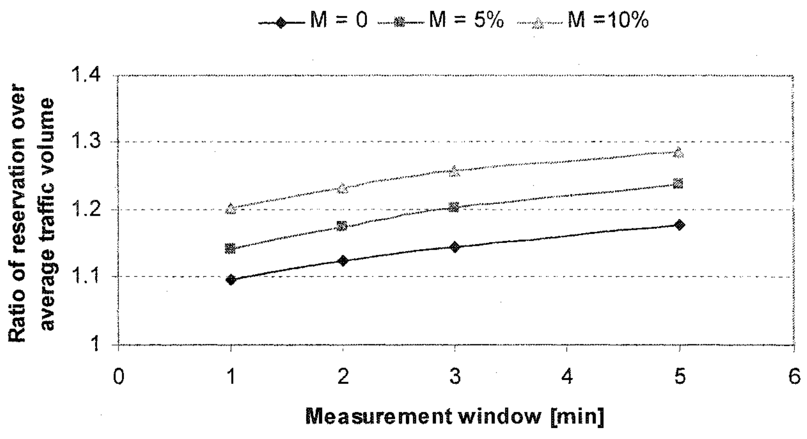

Figure 5-13. Ratio of reservation over average traffic volume vs. T_{meas} with buffer size of 10 packets when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

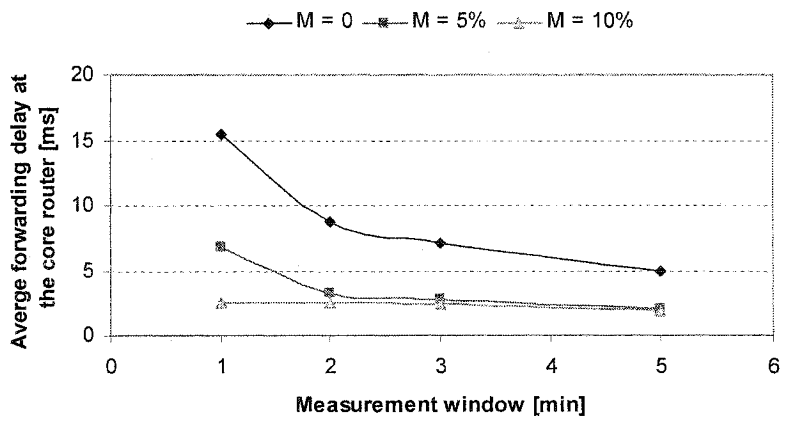

Figure 5-14. Average forwarding delay [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

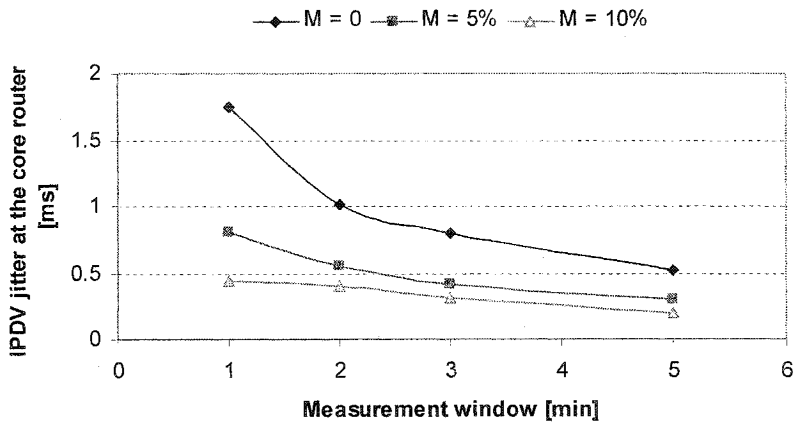

Figure 5-15. IPDV jitter [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

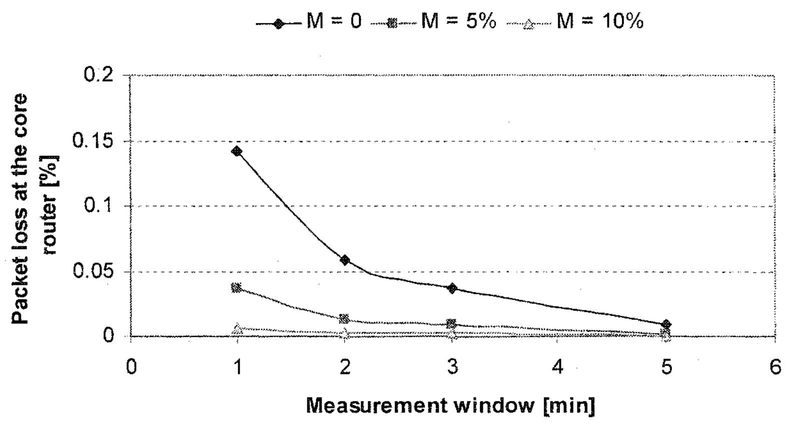

Figure 5-16a. The numerical value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

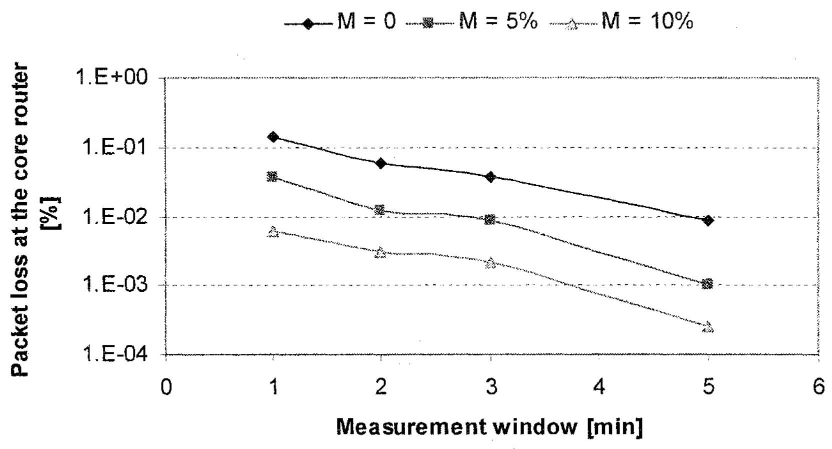

Figure 5-16b. The logarithm value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

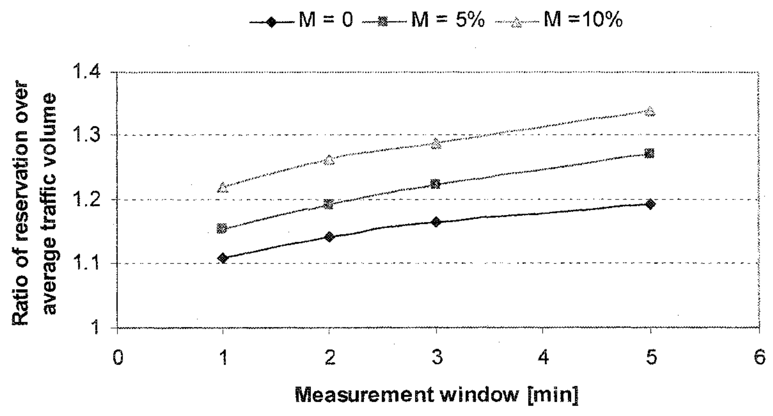

Figure 5-17. Ratio of reservation over average traffic volume vs. T_{meas} with buffer size of 10 packets when α–stable traffic with $\alpha = 1.60$ is applied and the ME is used

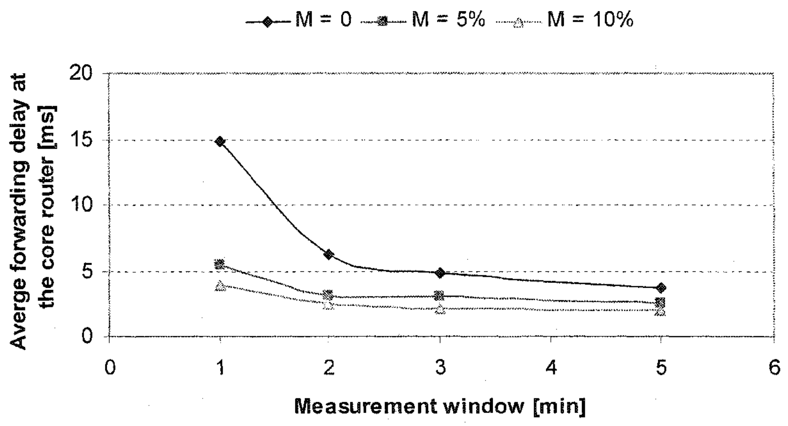

Figure 5-18. Average forwarding delay [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α–stable traffic with α = 1.60 is applied and the ME is used

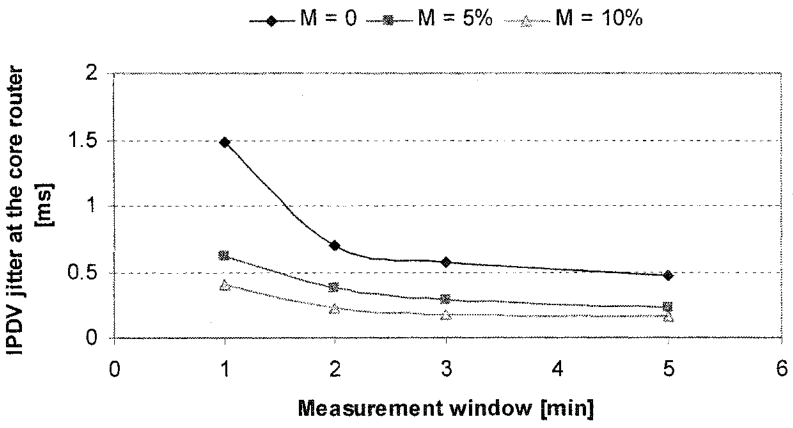

Figure 5-19. IPDV jitter [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α–stable traffic with α = 1.60 is applied and the ME is used

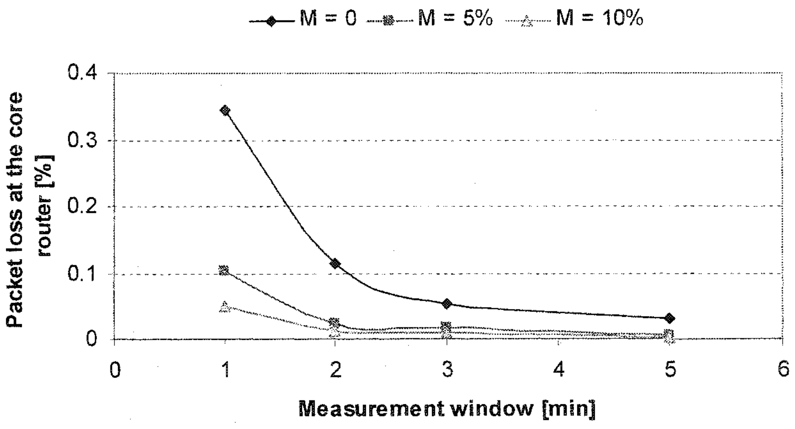

Figure 5-20a. The numerical value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α–stable traffic with α = 1.60 is applied and the ME is used

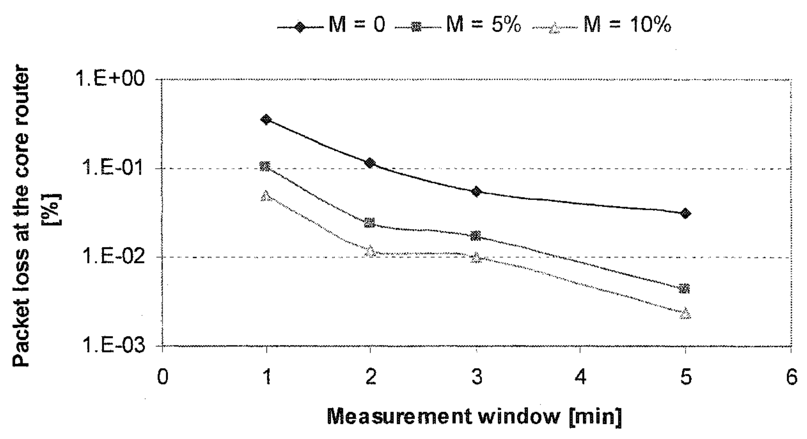

Figure 5-20b. The logarithm value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α–stable traffic with α = 1.60 is applied and the ME is used

Experiment 5-3

The objective of this set of experiments is to understand how the performance is affected by the value of Y (reservation window). $Y = N*X$ (N is the number of samples taken within the reservation window Y). In this case, we set $X = 5$ seconds, $T_{meas} = 5$ min, buffer size 10 packets, and run experiments with $Y = 1$ min, 2 min, 3 min and 5 min (N = 12, 24, 36 and 60). We apply margin $M = 0$. We plot the system's performance versus Y.

The value of Y controls the reservation interval. In this experiments, we choose $Y \leq T_{meas}$. Smaller Y means the frequency with which CBQ changes the allocation rate increase, it also means the signaling overhead is higher. Observe that the performance is practically unaffected by the value of reservation window Y, at least for the range of values examined (1 minute and higher). This is a positive outcome, since it indicates we can use quite long reservation window in order to reduce the signaling and processing overhead.

Please note that the conclusions we draw here refer to the case where the average traffic rate over a long period does not change. When a change in the average rate occurs (maybe because a new stream is added or another one is terminated), we wish to capture the change as fast as possible, in order to be able to "save" resource. If we react every 1 minute, we will be able to make the change in 1 minute, if we allocate resource for 5 minutes, we will end up having to stay for 5 minutes with the earlier resource. Similarly, if extra resource is needed, we will have massive losses for 5 minutes, and end up with termination of the application. Our results indicate there is no need for frequent change when the traffic does not change in term of average volume and statistics. However, the objective of our estimator is to be able to track such changes. If there is an increase in the average rate of the stream, the longer T_{meas} window we use, the longer might take to detect it. At the same time, while 2 to 3 minutes T_{meas} is enough for Poisson traffic, longer T_{meas} will help α–stable traffic. Thus, there is a trade-off to be taken into consideration. If the change of the average volume of traffic is gradual, we might wish to operate the longer T_{meas} window. If it is sudden, we might wish to be using shorter T_{meas} in order to be able to track it fast. Finally, the choice of Y is dependent on the limit of change we are placing on the traffic volume. If it is expected to be sudden and substantial, Y should be small, if not, large Y is fine.

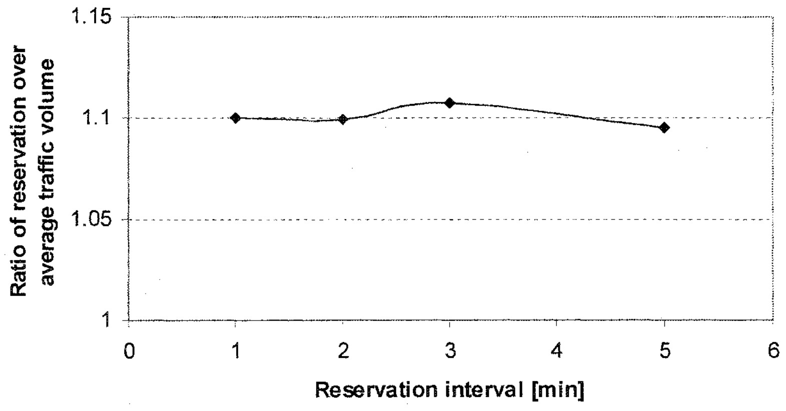

Figure 5-21. Ratio of reservation over average traffic volume vs. Y with buffer size of 10packet when Poisson traffic is applied and the ME is used

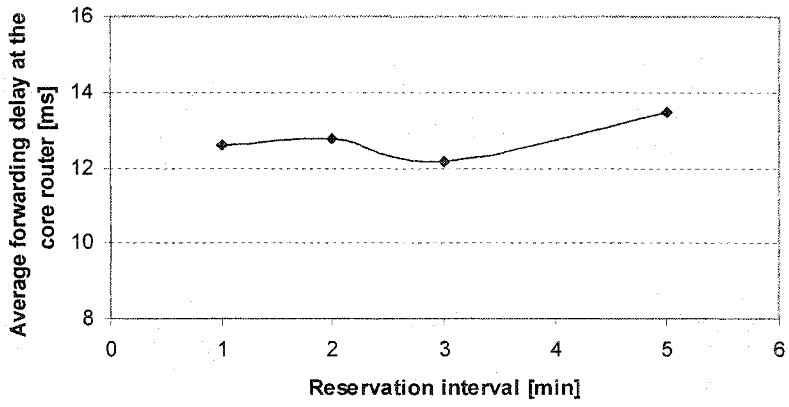

Figure 5-22. Average forwarding delay [ms] measured at the core router vs. *Y* with buffer size of 10packet when Poisson traffic is applied and the ME is used

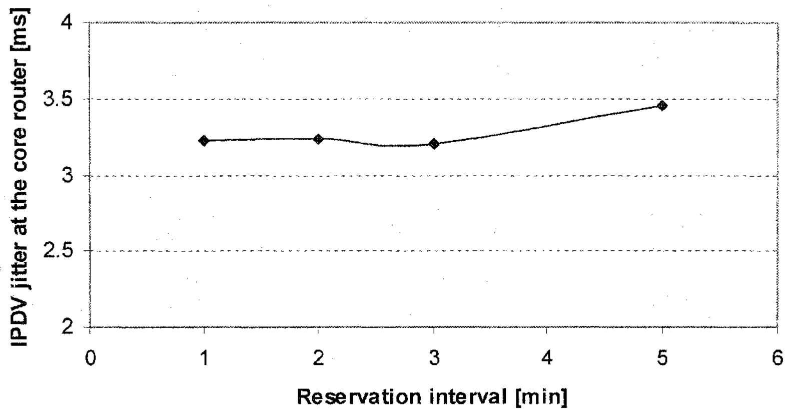

Figure 5-23. IPDV jitter [ms] measured at the core router vs. *Y* with buffer size of 10packet when Poisson traffic is applied and the ME is used

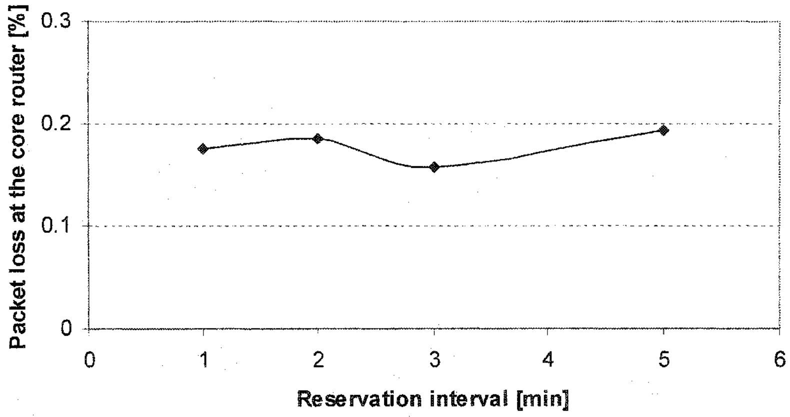

Figure 5-24a. The numerical value of Packet loss [%] measured at the core router vs. *Y* with buffer size of 10packet when Poisson traffic is applied and the ME is used

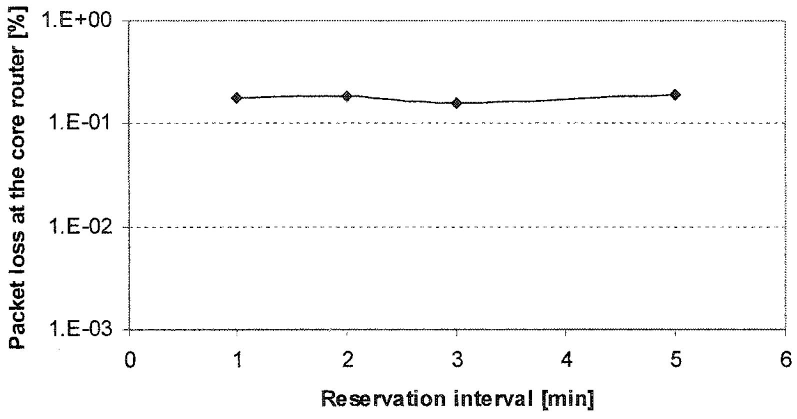

Figure 5-24b. The logarithm value of Packet loss [%] measured at the core router vs. *Y* with buffer size of 10packet when Poisson traffic is applied and the ME is used

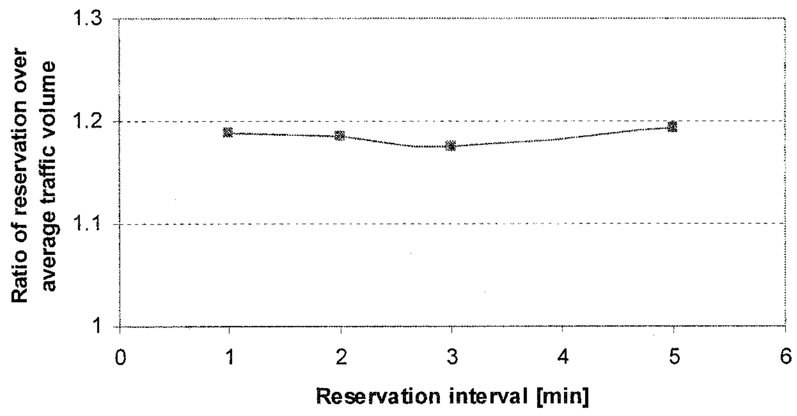

Figure 5-25. Ratio of reservation over average traffic volume vs. *Y* with buffer size of 10packet when α–stable traffic with α = 1.95 is applied and the ME is used

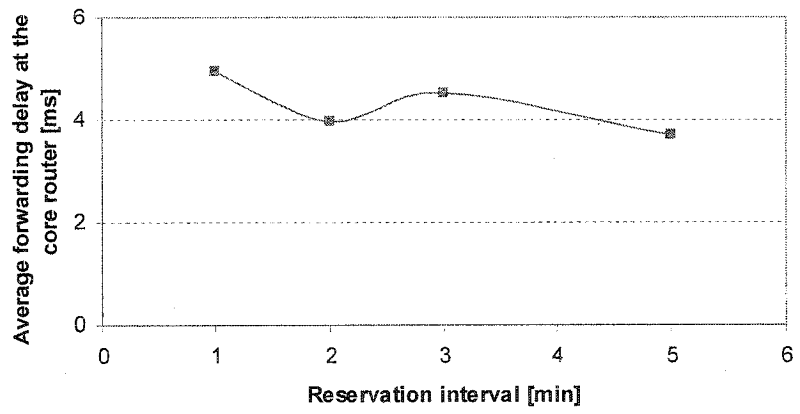

Figure 5-26. Average forwarding delay [ms] measured at the core router vs. *Y* with buffer size of 10packet when α–stable traffic with α = 1.95 is applied and the ME is used

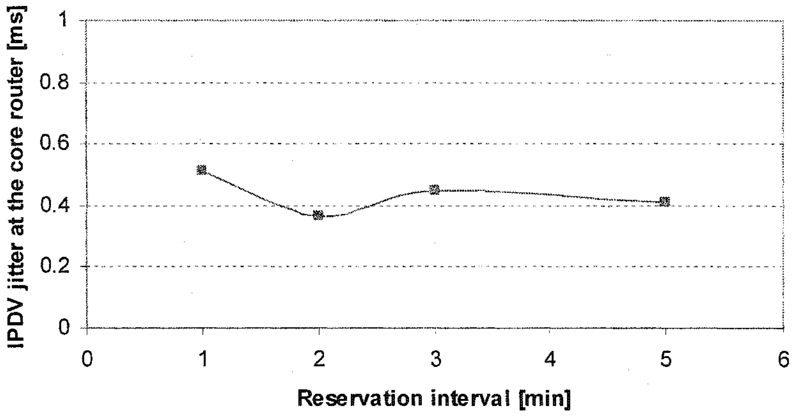

Figure 5-27. IPDV jitter [ms] measured at the core router vs. *Y* with buffer size of 10packet when α–stable traffic with α = 1.95 is applied and the ME is used

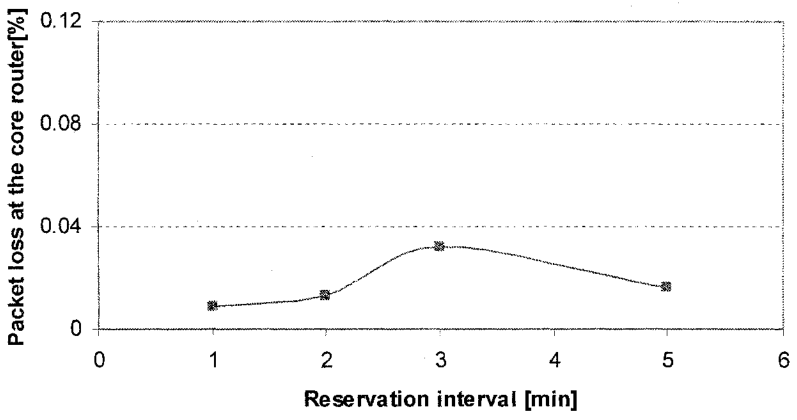

Figure 5-28a. The numerical value of Packet loss [%] measured at the core router vs. *Y* with buffer size of 10packet when α–stable traffic with α = 1.95 is applied and the ME is used

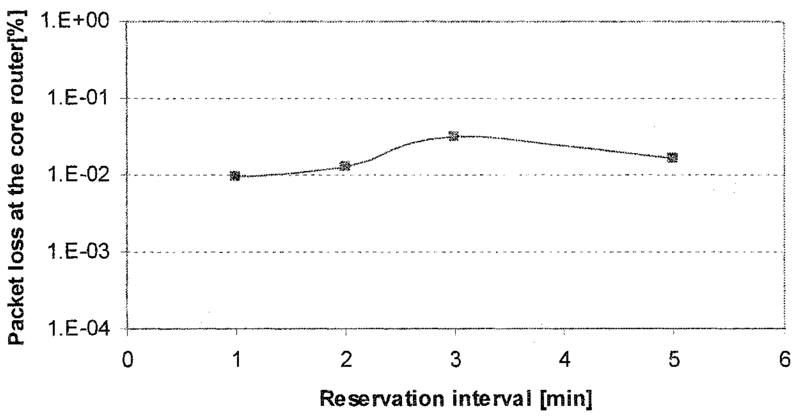

Figure 5-28b. The logarithm value of Packet loss [%] measured at the core router vs. *Y* with buffer size of 10packet when α–stable traffic with α = 1.95 is applied and the ME is used

Experiment 5-4

The objective of this set of experiments is to assess the effect of buffer size B on the performance. We choose a value of $X = 5$ sec, $T_{meas} = 5$ min and $Y = 1$ min. We run the ME with margin $M = 0$. Experiments are run for buffer size 5 packets, 10 packets, 50 packets and 100 packets.

Allocation rate has nothing to do with buffer size. However, buffering offers in a sense traffic smoothing. Smaller buffer size would result in high packet loss rate but lower delay. Larger buffer size will produce less loss rate but higher delay. Observe that the average forwarding delay increases, and the packet loss decreases when buffer size increases. But the IPDV jitter seems not to be sensitive to buffer size. Several interesting observations can be made based on the results reported here.

The average forwarding delay for α–stable traffic remains lower than Poisson when small buffer size is applied, however, it grows higher for larger buffer size. The reason for this behavior is as follows. α–stable traffic tends to be producing more intensive, longer bursts as compared to Poisson. When such bursts occur and the buffer size is small, it results in dropping of packets, which do not count in delay. At the same time, α–stable traffic has a longer period of low activity as compared to Poisson. During this period, packets go through fast. Thus, the delay for α–stable traffic when having small buffer size is smaller than Poisson. When the buffer becomes larger, the long bursts are absorbed by the buffer and the packets accumulate considerable delay values. As consequence, the average delay for α–stable traffic experiences longer average delays when large buffer size is used.

Observe that IPDV for α–stable traffic is lower than Poisson for the same reason as indicated before. Another interesting observation is that IPDV seems to be unaffected by the buffer size. The possible reason is that IPDV is not the deviation of the delay from the average value. It is the difference of delays experienced by consecutive packets. Close-by packets tend to experience the same delays, since they experience the same network condition, thus their difference remains small and unaffected by the buffer size.

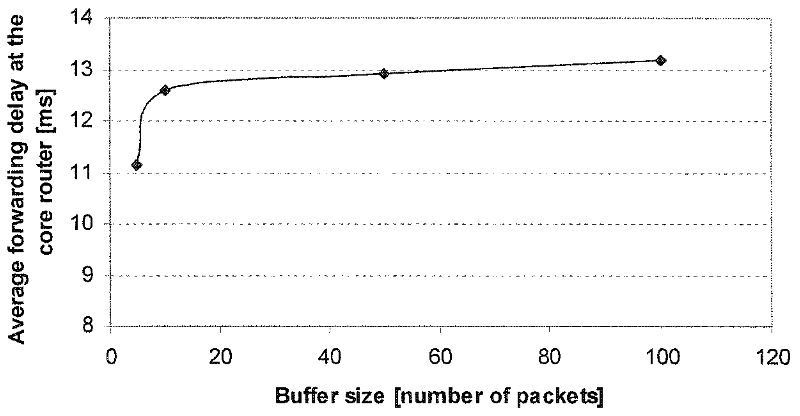

Figure 5-29. Average forwarding delay [ms] measured at the core router vs. *B* when Poisson traffic is applied and the ME is used

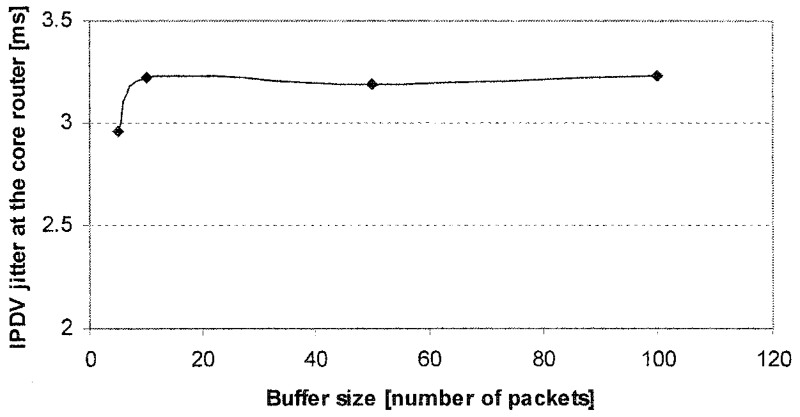

Figure 5-30. IPDV jitter [ms] measured at the core router vs. *B* when Poisson traffic is applied and the ME is used

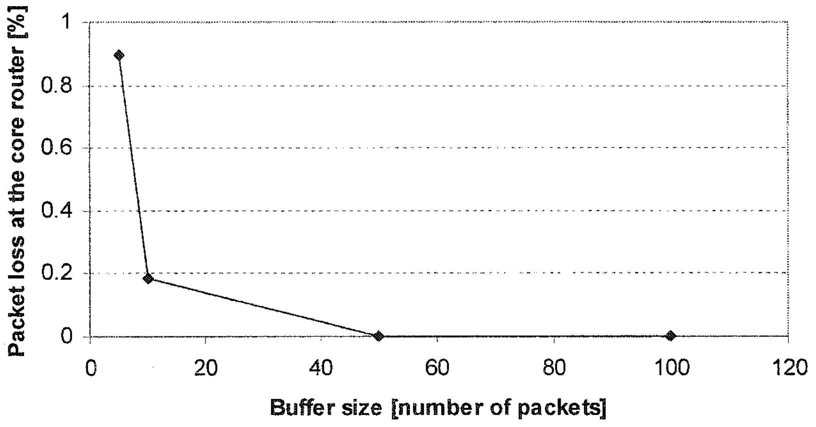

Figure 5-31a. The numerical value of packet loss [%] measured at the core router vs. *B* when Poisson traffic is applied and the ME is used

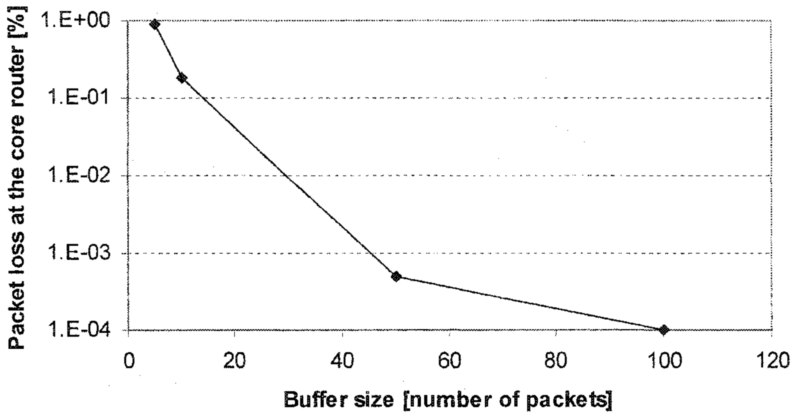

Figure 5-31b.The logarithm value of packet loss [%] measured at the core router vs. *B* when Poisson traffic is applied and the ME is used

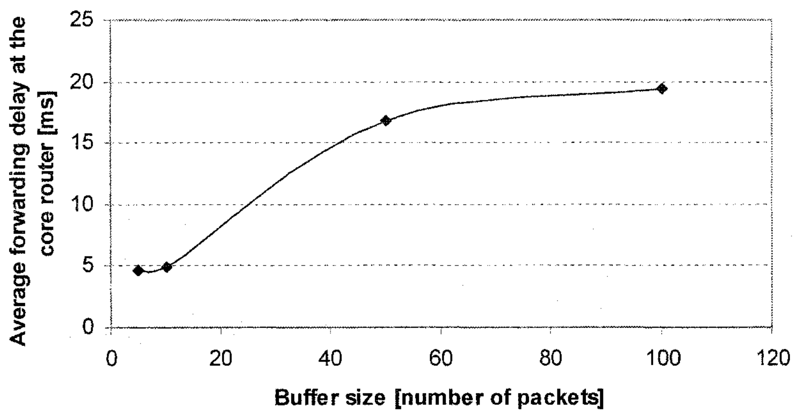

Figure 5-32. Average forwarding delay [ms] measured at the core router vs. *B* when α–stable traffic with α = 1.95 is applied and the ME is used

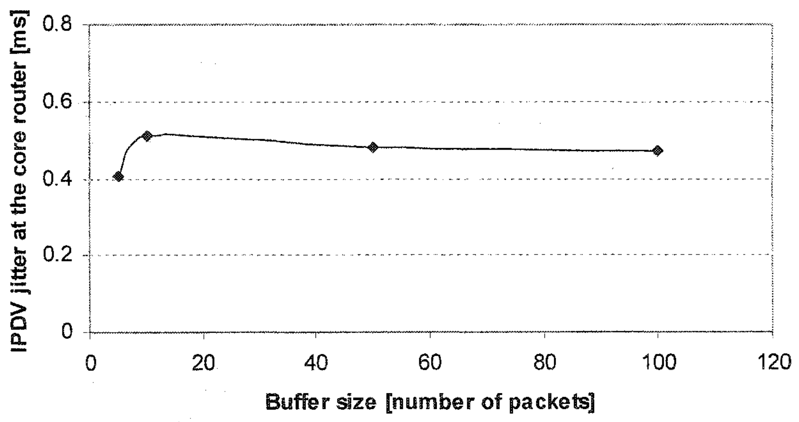

Figure 5-33. IPDV jitter [ms] measured at the core router vs. *B* when α–stable traffic with α = 1.95 is applied and the ME is used

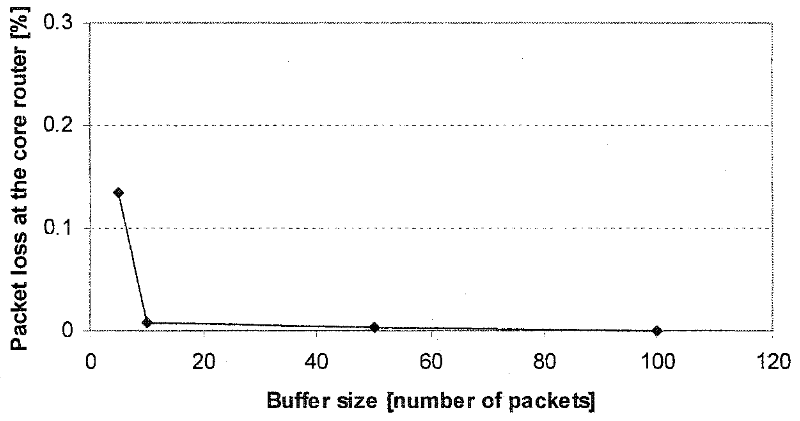

Figure 5-34a. The numerical value of packet loss [%] measured at the core router vs. B when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

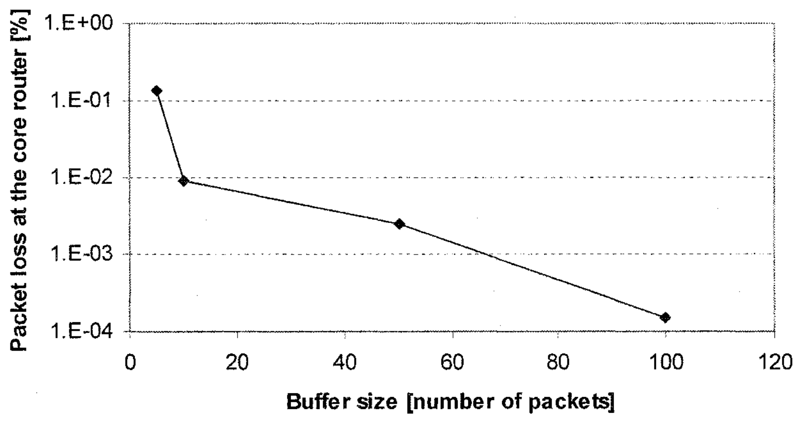

Figure 5-34b. The logarithm value of packet loss [%] measured at the core router vs. B when α–stable traffic with $\alpha = 1.95$ is applied and the ME is used

5.3 EVALUATION OF GAUSSIAN ESTIMATOR (GE)

Let us recall the GE operates as follows:

$$R_{ren} = m + a\sqrt{v}$$

where R_{ren} is the renegotiated rate, m and v are respectively the mean and variance of the rate samples R_i during the period of measurement window, a is a scale factor that controls the extent to which the negotiated rate accommodates variability in the samples. In our implementation, the value of a is controlled by choosing different k value (refer to Section 3.1.3 for the meaning of k). There is mapping between k and a (refer to Table 3-1). For example, when we choose $k = 1$, the probability of packet loss $p = 10^{-k} = 0.1$, the corresponding $a = 1.2815$. The bigger k is, the bigger a becomes. For the sake of simplicity, the margin size is controlled by the value of k. We repeat the experiments for GE. Both Poisson and α–stable traffic are evaluated.

Experiment 5-5

We repeat Experiment 5-1 but using GE instead of the ME. We use the same parameters as indicated earlier to assess the effect of X on the performance. We set $Y = 1$ min and $T_{meas} = 5$ min. We apply a buffer size of 10 packets and 100 packets at the core router to run experiments using $k = 2$ (refer to Section 3.1.3) for X =: 5 seconds, 10 seconds, 15 seconds and 20 seconds. We plot the performance versus X.

Observe that the GE shows similar behavior as the ME in Experiment 5-1. As we indicated earlier for the ME case, the traffic samples produced when processing a traffic stream with larger X, become "smoother" (they show less deviation from average) as compared to when they are produced with small X. The variance of smoother process decreases, thus, the allocation rate decreases when the value of X increases.

However, there are some additional observations to be made. For Poisson traffic, the results of GE and ME are very close in all cases. This could be attributed to the nature of Poisson traffic and the fact it is a memoryless process. In both cases, the estimate made by the estimators based on measurements conducted within a time window, do not provide any reduction in the uncertainty produced by the randomness of the traffic within the following time window.

For α-stable traffic, the situation is different. The GE outperforms the ME in all cases where the value of X happens to be larger than 5 seconds (for the 5 seconds case, the results seem somehow similar). This behavior can be attributed to the fact that while the ME makes reservations based on the maximum average value calculated within several time intervals X, the GE takes the variability, the values of these intervals demonstrated, into consideration.

As the size of X increases, the values of the sample rate show stronger tendency to fall closer to the average traffic value. Thus, they gradually lose the ability to "reflect " the burstiness level of the traffic. As the GE takes into consideration the deviations of these values, it is capable of identifying burstiness to some extent even for higher X intervals, thus requests or releases resources more successfully than the ME.

By inspecting the figures for average delay and packet losses, the superiority of GE at higher values X as compared to the ME is evident.

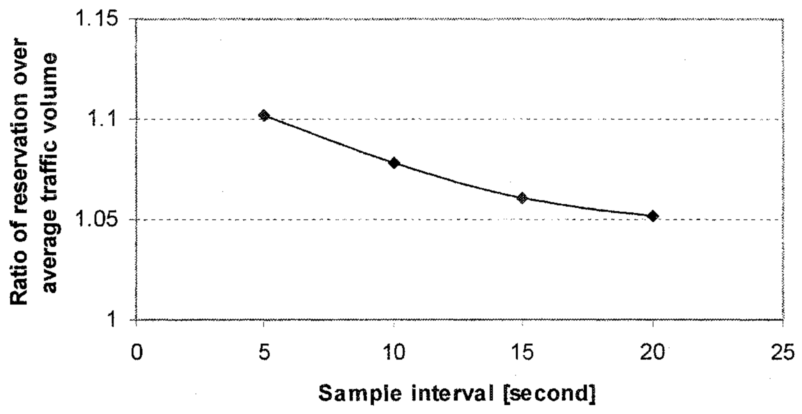

Figure 5-35. Ratio of reservation over average traffic volume vs. X when Poisson traffic is applied and the GE is used

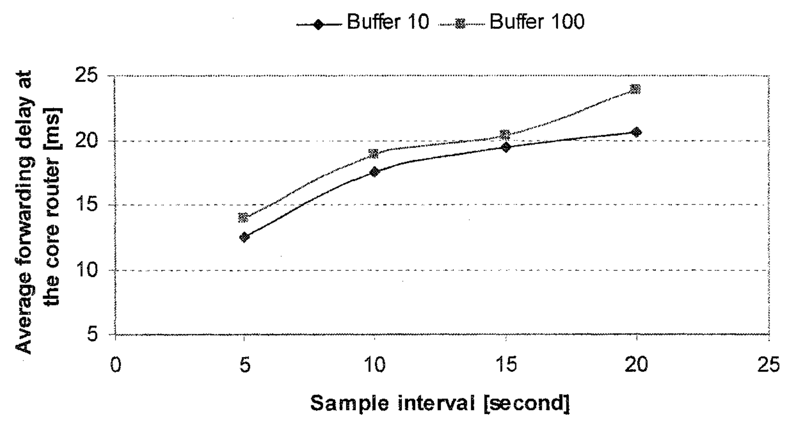

Figure 5-36. Average forwarding delay [ms] measured at the core router vs. X with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and the GE is used

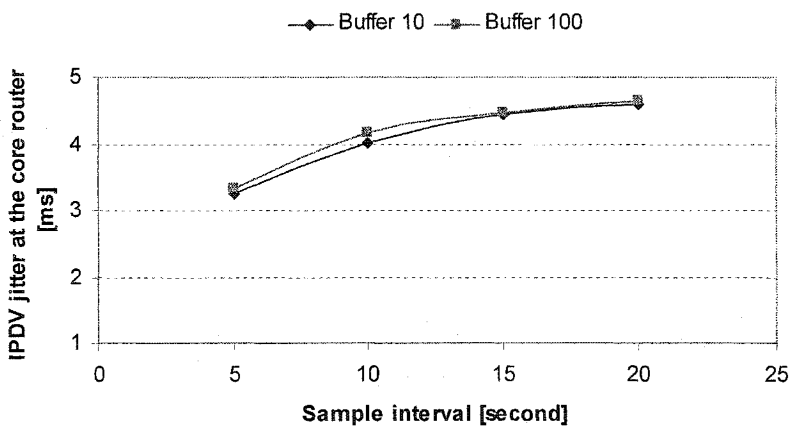

Figure 5-37. IPDV jitter [ms] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and the GE is used

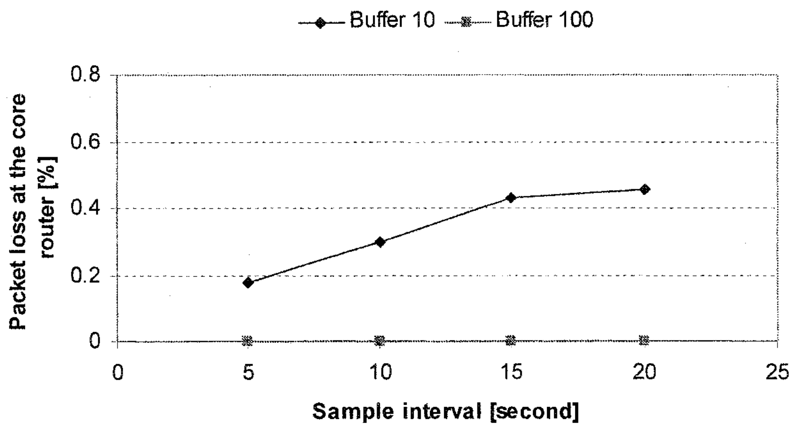

Figure 5-38a. The numerical value of packet loss [%] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and the GE is used

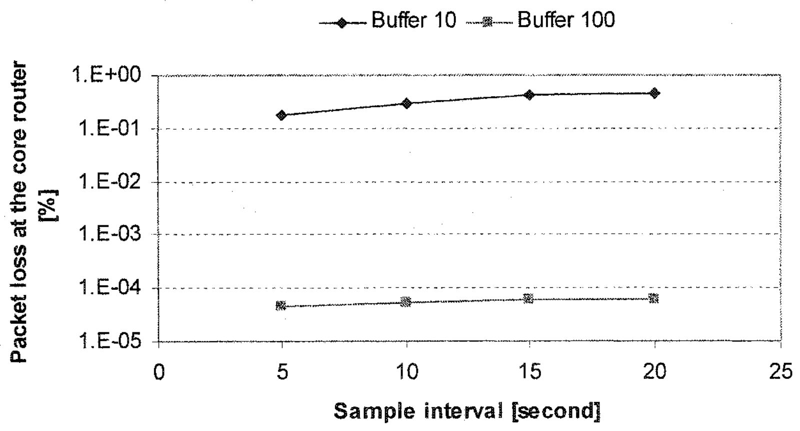

Figure 5-38b. The logarithm value of packet loss [%] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and the GE is used

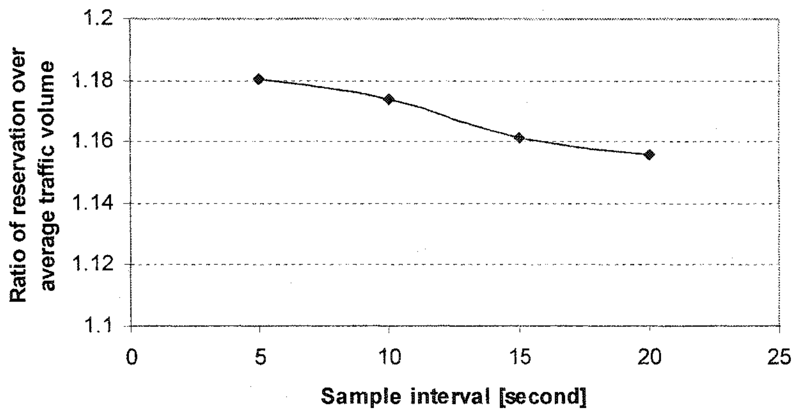

Figure 5-39. Ratio of reservation over average traffic volume vs. X when α–stable traffic with $\alpha = 1.95$ is applied and the GE is used

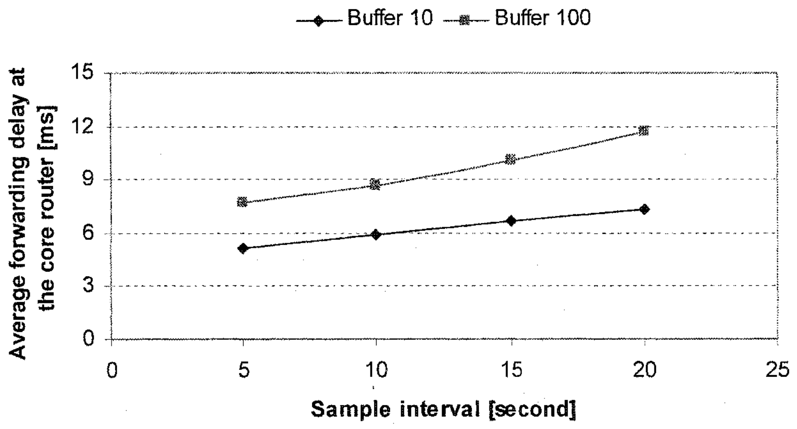

Figure 5-40. Average forwarding delay [ms] measured at the core router vs. X with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with $\alpha = 1.95$ is applied and the GE is used

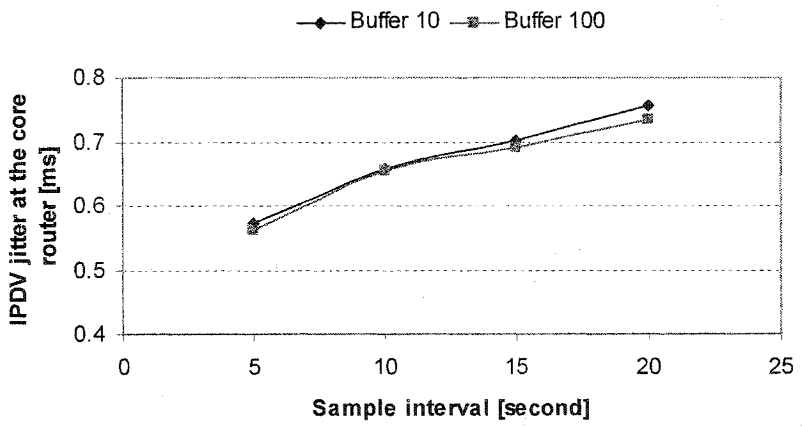

Figure 5-41. IPDV jitter [ms] measured at the core router vs. X with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with $\alpha = 1.95$ is applied and the GE is used

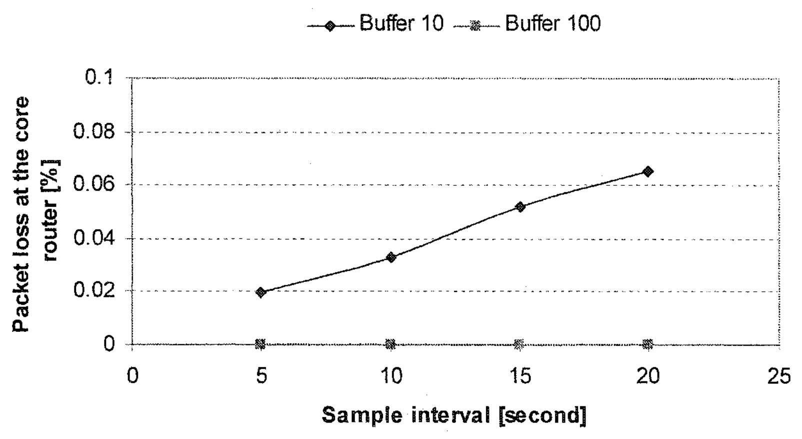

Figure 5-42a. The numerical value of packet loss [%] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with $\alpha = 1.95$ is applied and the GE is used

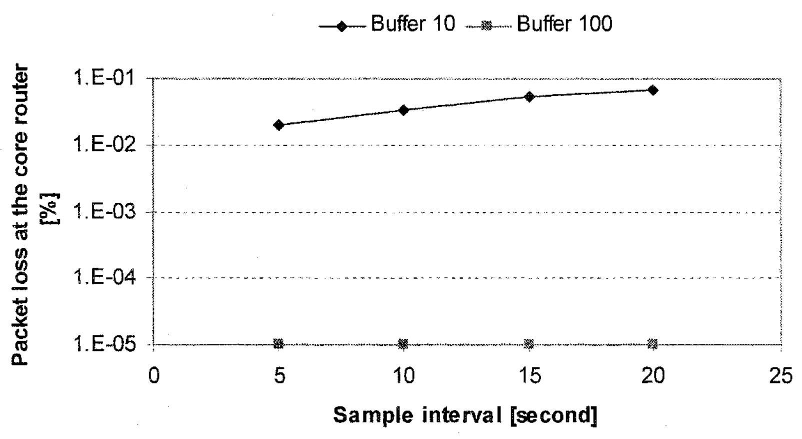

Figure 5-42b. The logarithm value of packet loss [%] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with $\alpha = 1.95$ is applied and the GE is used

Experiment 5-6

We repeat Experiment 5-2 with GE. We use the same parameters as indicated earlier to assess the effect of T_{meas} and margin size on the performance. Here the margin is controlled by $k = 1, 2$ and 3. The bigger the value of k is, the bigger the value of scale factor a is.

The GE displays similar behavior with the ME. Comparing the results, it appears that in terms of performance, the $M = 0$ case of ME and $k = 2$ case of GE are the ones that are close to each other. As expected, $k = 1$ gives high packet loss rate and has poor performance.

For Poisson traffic, the reservation of GE is remarkably flat. This is due to the memoryless nature of the Poisson process. As long as the window T_{meas} is long enough to give good statistics, the estimates of average and variance do not change. On the contrary, when the observation window T_{meas} increases, the probability of getting an isolated larger sample value increases. For α–stable traffic, the reservation tends to increase as the T_{meas} increases, even though the level of increase is still smaller than that of ME. This behavior should be contributed to the nature of α–stable traffic. α–stable traffic has infinite variance. Of course, a truncated α–stable process (as is the case we encounter since we are using a T_{meas} window) shows finite variance behavior, however the variance increases with the increase of T_{meas}. As $T_{meas} \to \infty$, the variance will approach ∞.

It appears a T_{meas} = 2 min is adequate to reach the best level of performance possible.

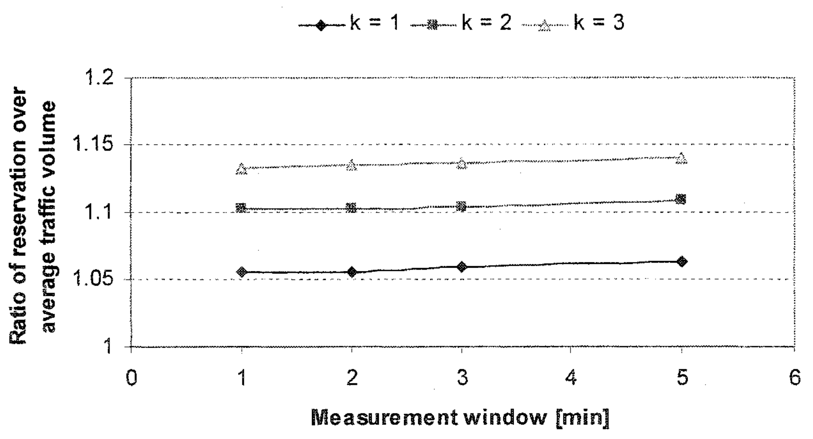

Figure 5-43. Ratio of reservation over average traffic volume vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the GE is used

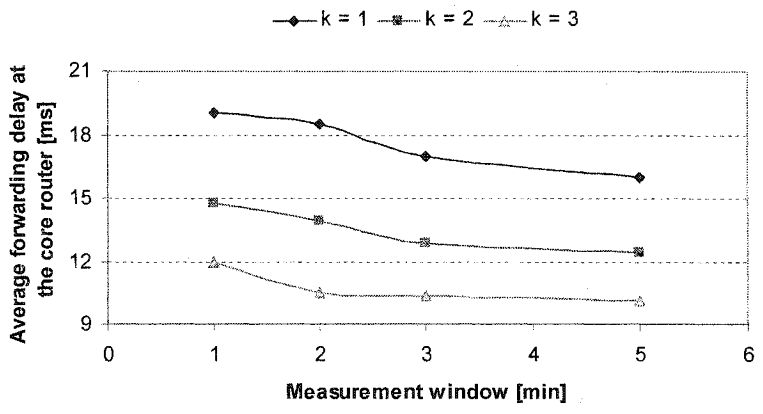

Figure 5-44. Average forwarding delay [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the GE is used

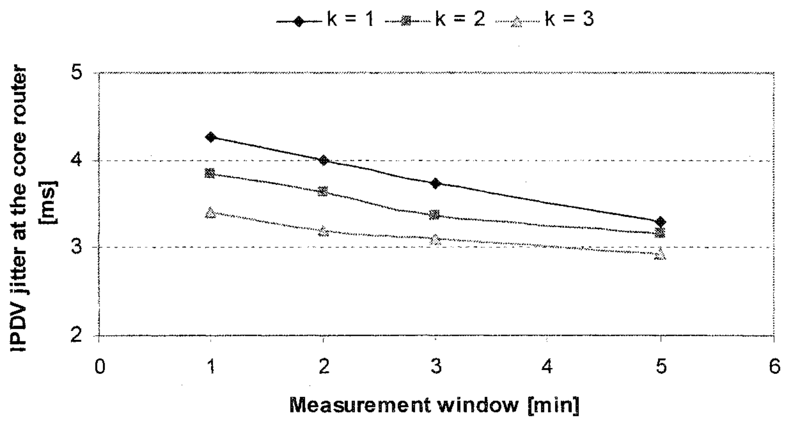

Figure 5-45. IPDV jitter [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the GE is used

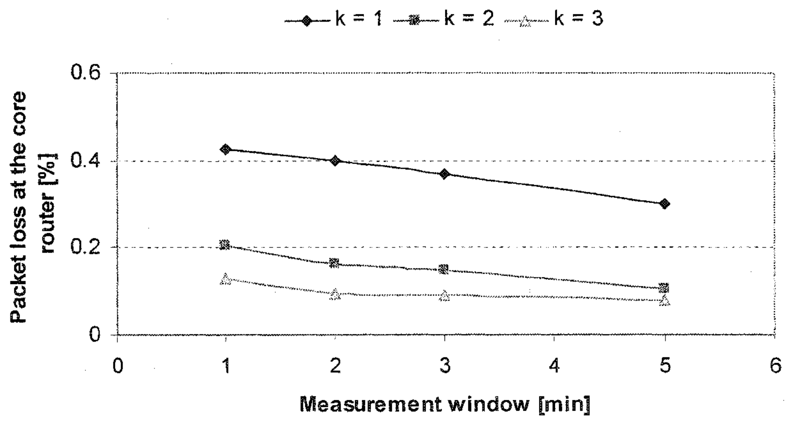

Figure 5-46a. The numerical value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the GE is used

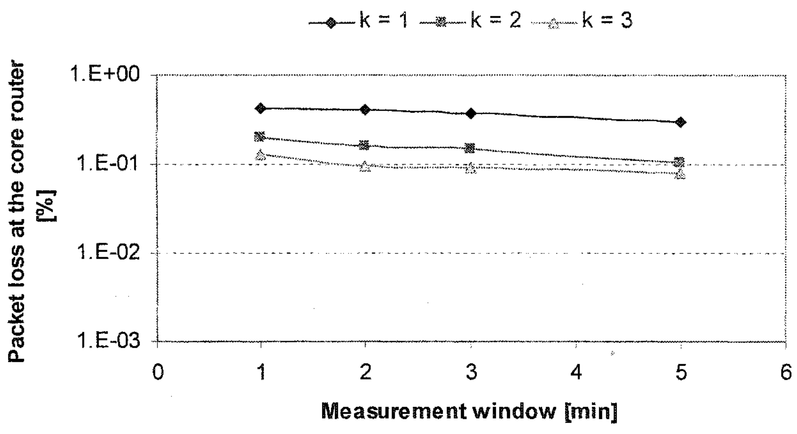

Figure 5-46b. The logarithm value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the GE is used

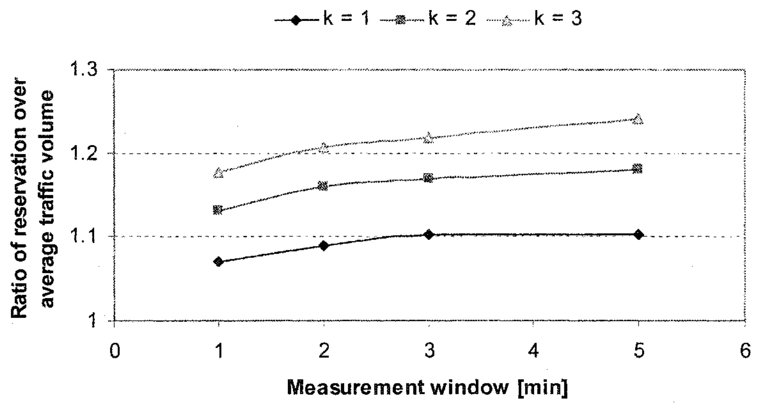

Figure 5-47.Ratio of reservation over average traffic volume vs. T_{meas} with buffer size of 10 packets when α-stable traffic with $\alpha = 1.95$ is applied and the GE is used

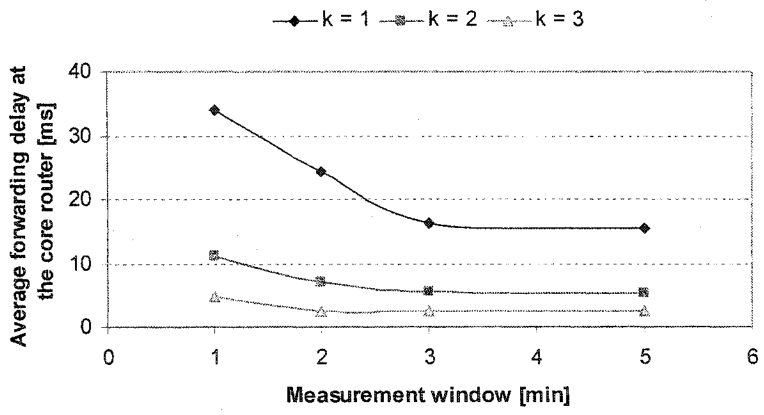

Figure 5-48. Average forwarding delay [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with $\alpha = 1.95$ is applied and the GE is used

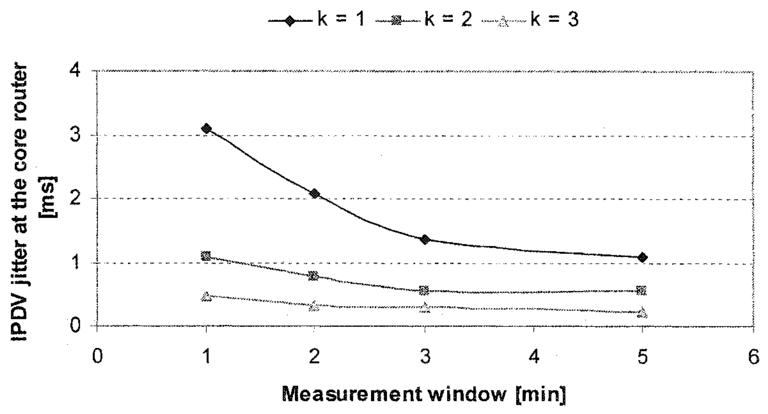

Figure 5-49. IPDV jitter [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with α = 1.95 is applied and the GE is used

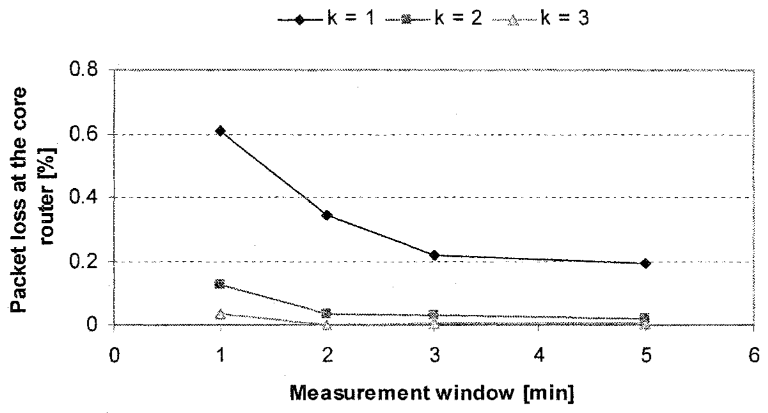

Figure 5-50a. The numerical value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with α = 1.95 is applied and the GE is used

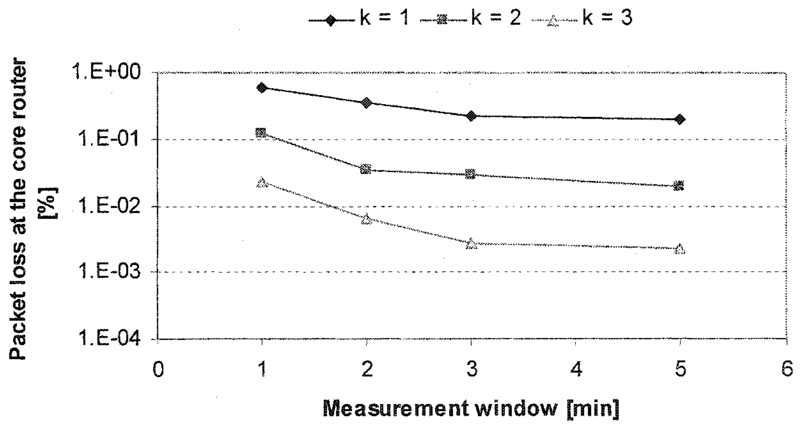

Figure 5-50b. The logarithm value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with α = 1.95 is applied and the GE is used

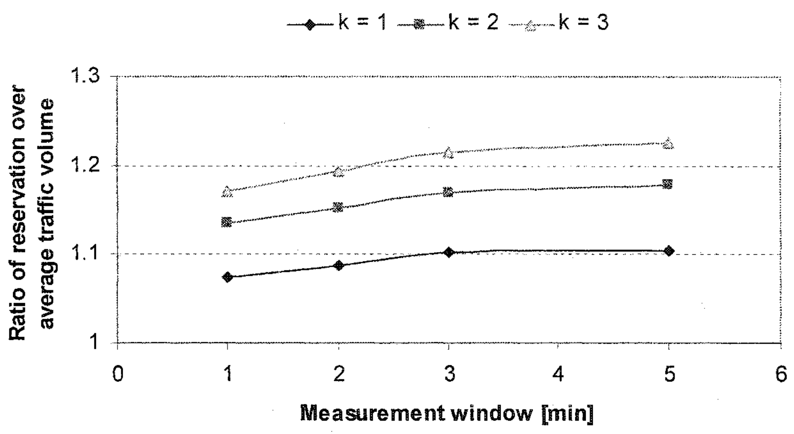

Figure 5-51. Ratio of reservation over average traffic volume vs. T_{meas} with buffer size of 10 packets when α-stable traffic with $\alpha = 1.60$ is applied and the GE is used

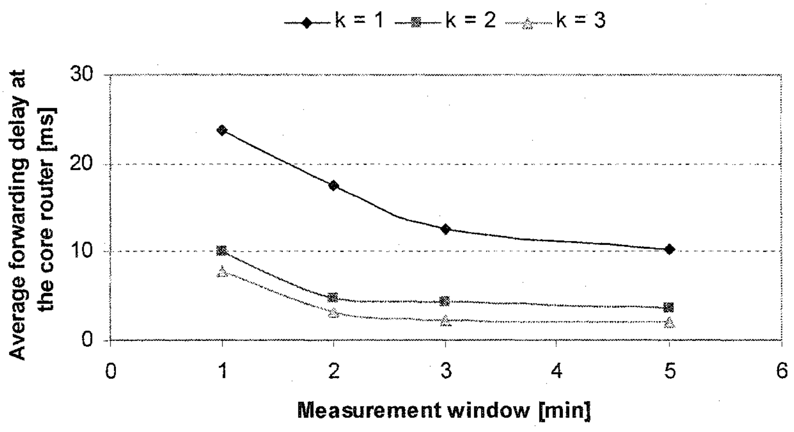

Figure 5-52. Average forwarding delay [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with $\alpha = 1.60$ is applied and the GE is used

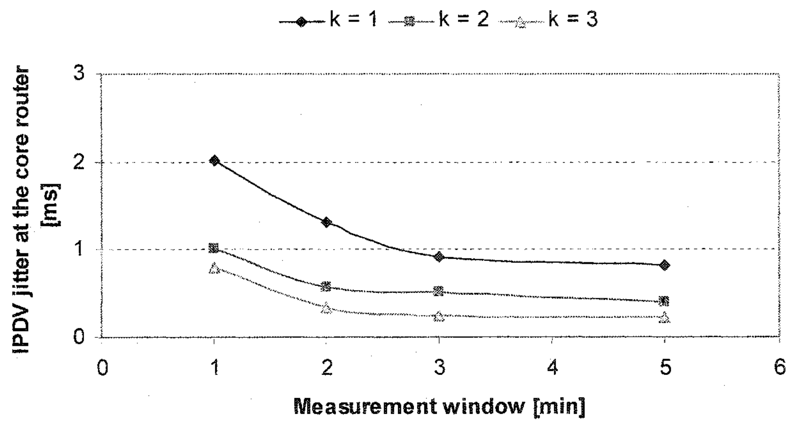

Figure 5-53. IPDV jitter [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with $\alpha = 1.60$ is applied and the GE is used

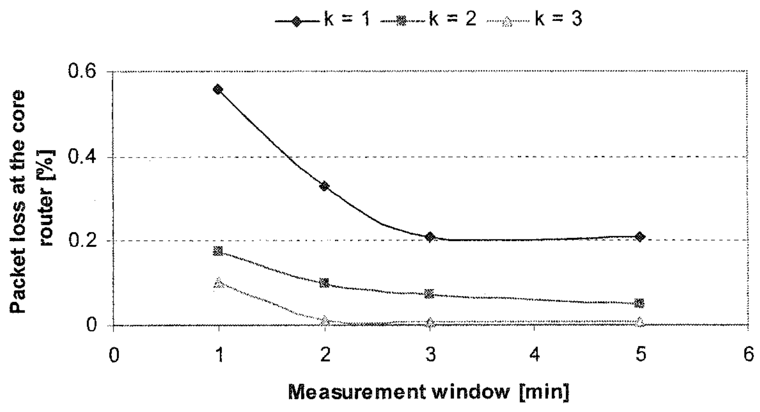

Figure 5-54a. The numerical value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with α = 1.60 is applied and the GE is used

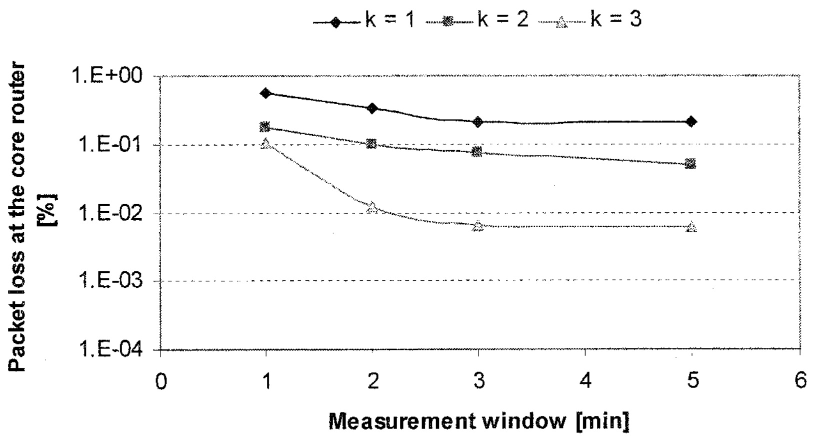

Figure 5-54b. The logarithm value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with α = 1.60 is applied and the GE is used

Experiment 5-7

We repeat Experiment 5-3 using GE. We use the same parameters used earlier in Experiment 5-3 to assess the effect of Y on the performance. The set parameters are $X = 5$ seconds, $T_{meas} = 5$ min, buffer size $B = 10$ packets, $k = 2$. We run experiments with $Y = 1$ min, 2 min, 3 min and 5 min ($N = 12, 24, 36, 60$).

Observe that GE shows similar behavior with ME. It is quite evident that the value of Y does not have any serious performance impact both for Poisson and α–stable traffic.

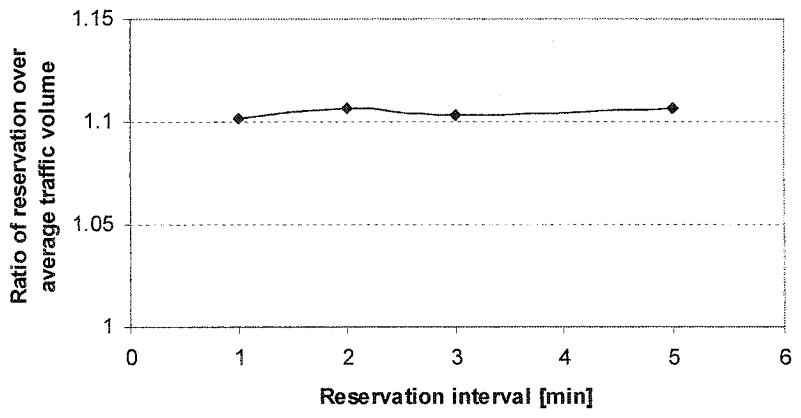

Figure 5-55. Ratio of reservation over average traffic volume vs. *Y* when Poisson traffic is applied and the GE is used

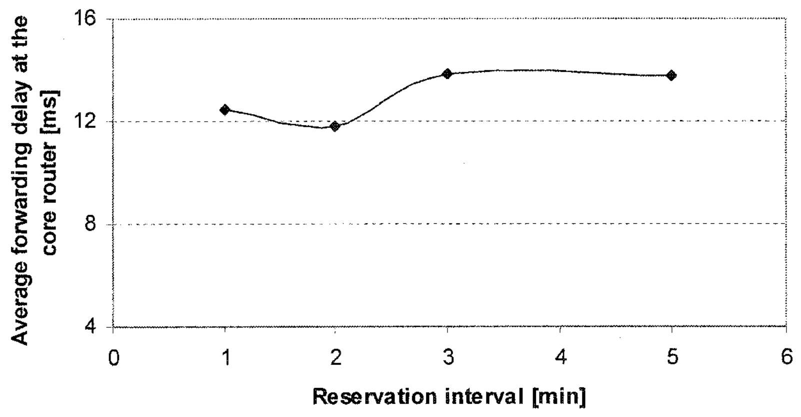

Figure 5-56. Average forwarding delay [ms] measured at the core router vs. *Y* with buffer size of 10 packets when Poisson traffic is applied and the GE is used

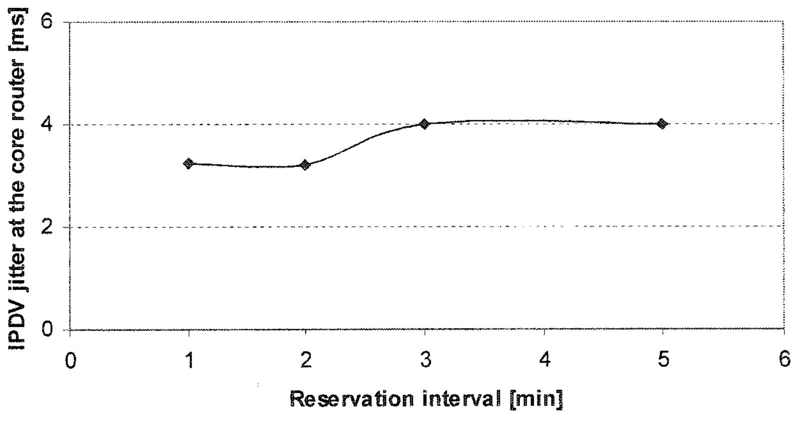

Figure 5-57. IPDV jitter [ms] measured at the core router vs. *Y* with buffer size of 10 packets when Poisson traffic is applied and the GE is used

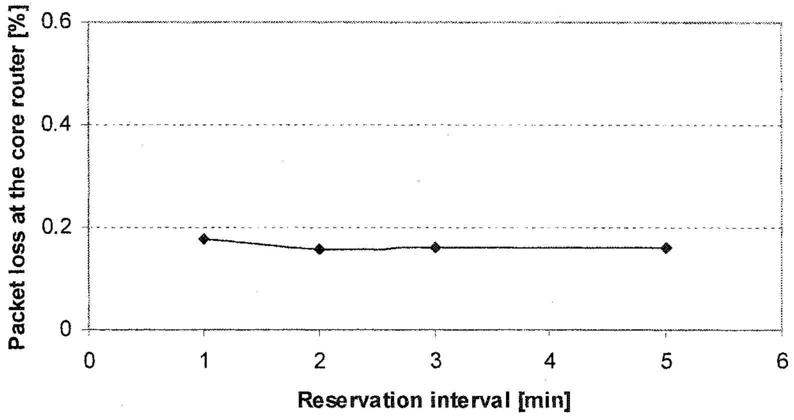

Figure 5-58a. The numerical value of packet loss [%] measured at the core router vs. *Y* with buffer size of 10 packets when Poisson traffic is applied and the GE is used

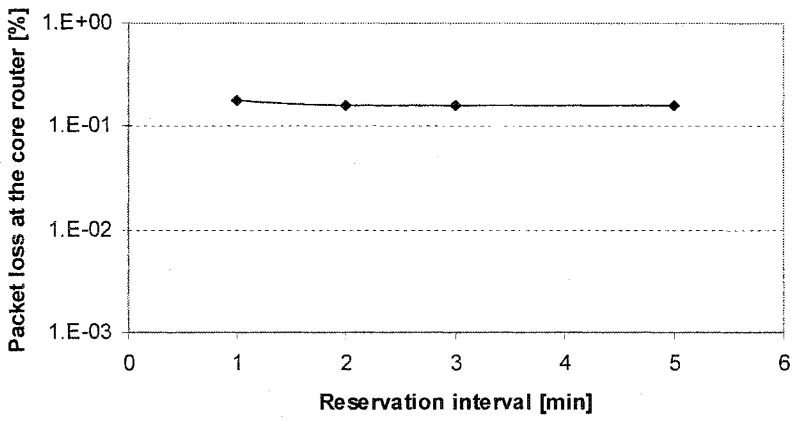

Figure 5-58b. The logarithm value of packet loss [%] measured at the core router vs. *Y* with buffer size of 10 packets when Poisson traffic is applied and the GE is used

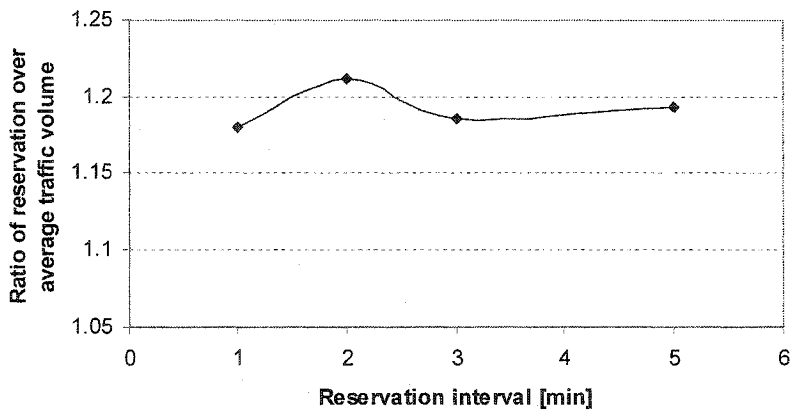

Figure 5-59. Ratio of reservation over average traffic volume vs. *Y* with buffer size of 10 packets when α–stable traffic with $\alpha = 1.95$ is applied and the GE is used

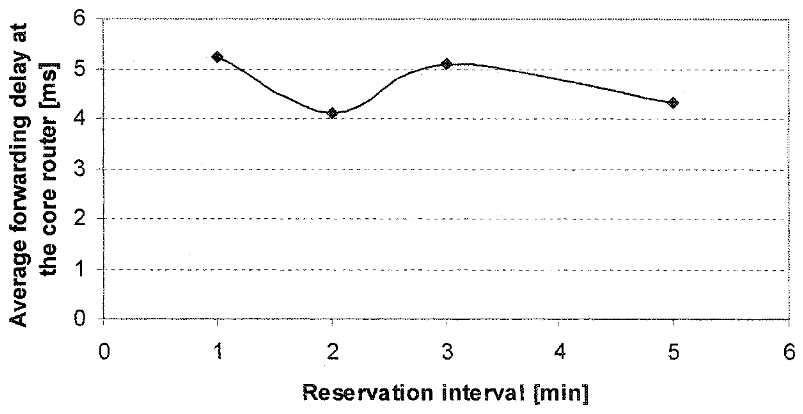

Figure 5-60. Average forwarding delay [ms] measured at the core router vs. Y with buffer size of 10 packets when α–stable traffic with $\alpha = 1.95$ is applied and the GE is used

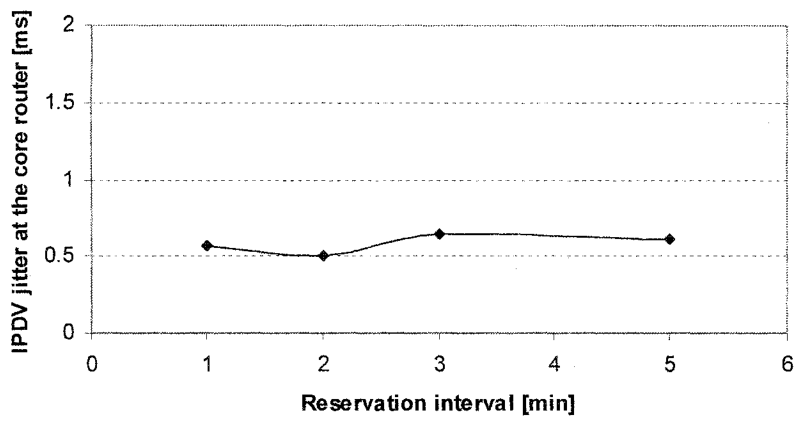

Figure 5-61. IPDV jitter [ms] measured at the core router vs. Y with buffer size of 10 packets when α–stable traffic with $\alpha = 1.95$ is applied and the GE is used

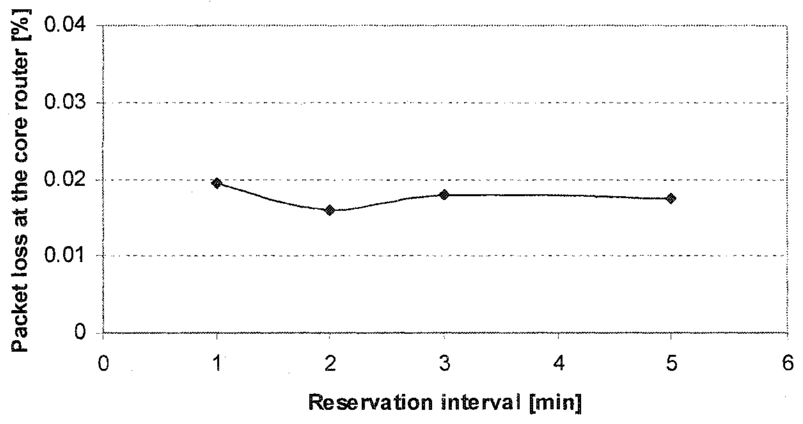

Figure 5-62a. The numerical value of packet loss [%] measured at the core router vs. Y with buffer size of 10 packets when α–stable traffic with $\alpha = 1.95$ is applied and the GE is used

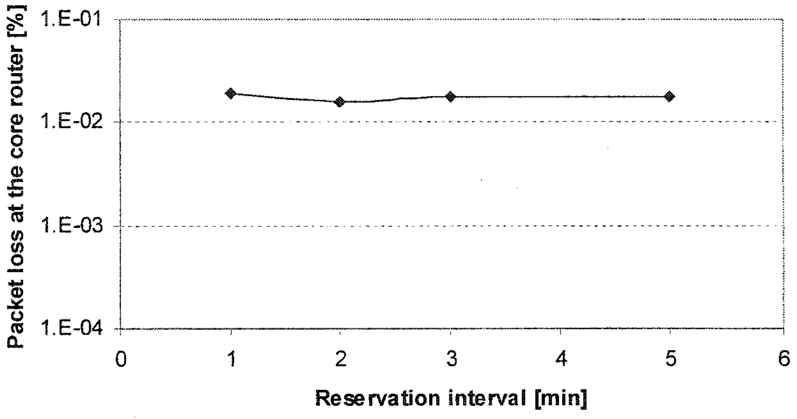

Figure 5-62b. The logarithm value of packet loss [%] measured at the core router vs. *Y* with buffer size of 10 packets when α–stable traffic with α = 1.95 is applied and the GE is used

Experiment 5-8

We repeat Experiment 5-4 using the GE. We use the same parameters as in Experiment 5-4 to assess the effect of buffer size *B*. We set $X = 5$ seconds, $T_{meas} = 5$ min and $Y = 1$ min and run GE with $k = 2$. The buffer size used to run the experiments are 5 packets, 10 packets, 50 packets and 100 packets.

Observe that the packet loss is reduced, the average forwarding delay increases but the IPDV jitter changes smoothly when the buffer size increases. This occurs for both Poisson and α–stable traffic. In summary, the results for both ME and GE appear to be very close.

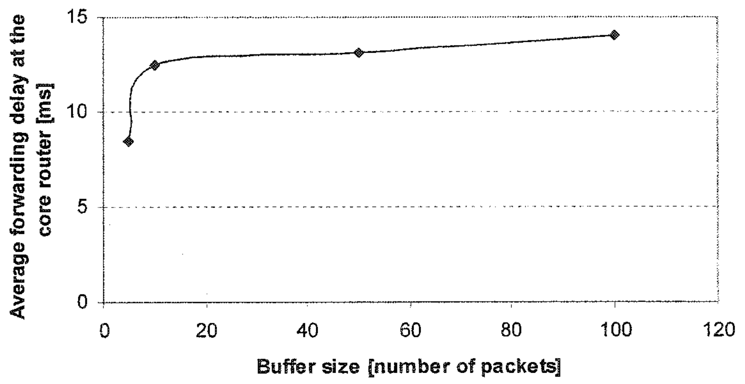

Figure 5-63. Average forwarding delay [ms] measured at the core router vs. *B* when Poisson traffic is used and the GE is used

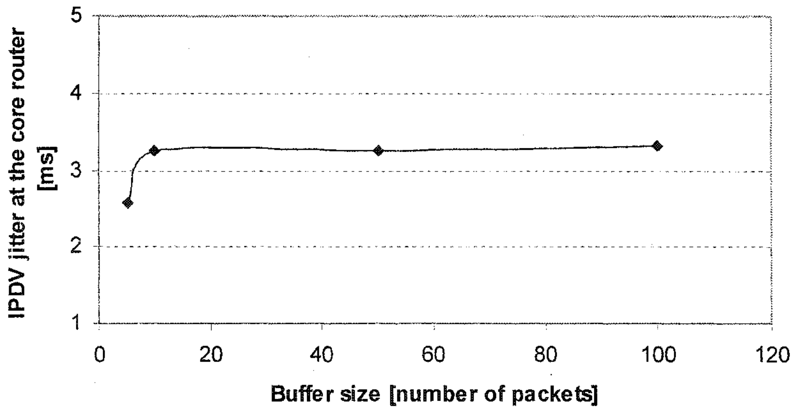

Figure 5-64. IPDV jitter [ms] measured at the core router vs. *B* when Poisson traffic is used and the GE is used

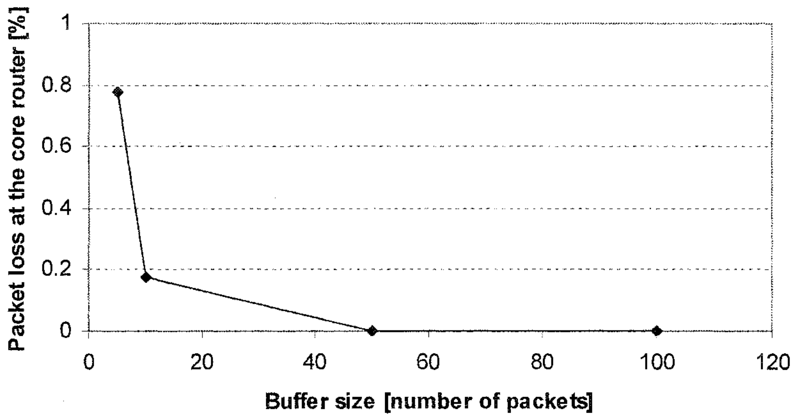

Figure 5-65a. The numerical value of packet loss [%] measured at the core router vs. *B* when Poisson traffic is used and the GE is used

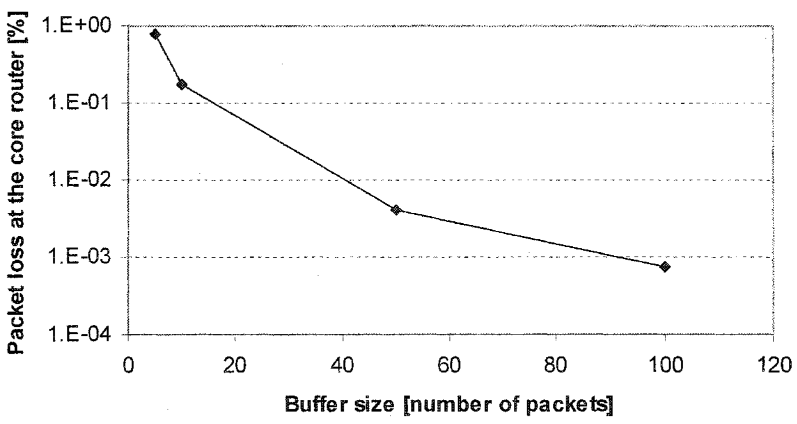

Figure 5-65b. The logarithm value of packet loss [%] measured at the core router vs. *B* when Poisson traffic is used and the GE is used

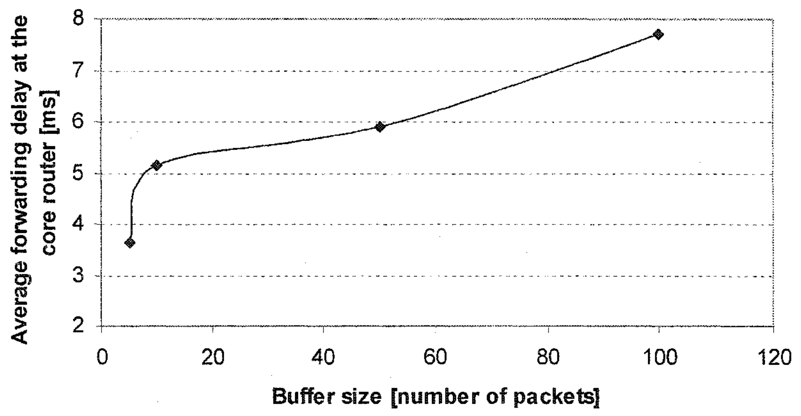

Figure 5-66. Average forwarding delay [ms] measured at the core router vs. B when α–stable traffic with α = 1.95 is used and the GE is used

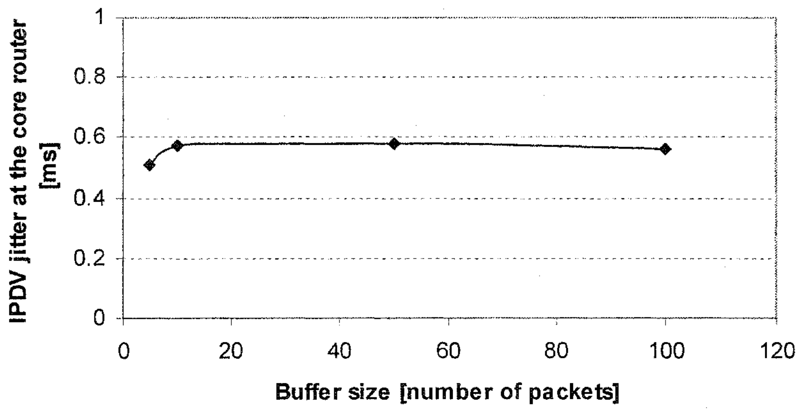

Figure 5-67. IPDV jitter [ms] measured at the core router vs. B when α–stable traffic with α = 1.95 is used and the GE is used

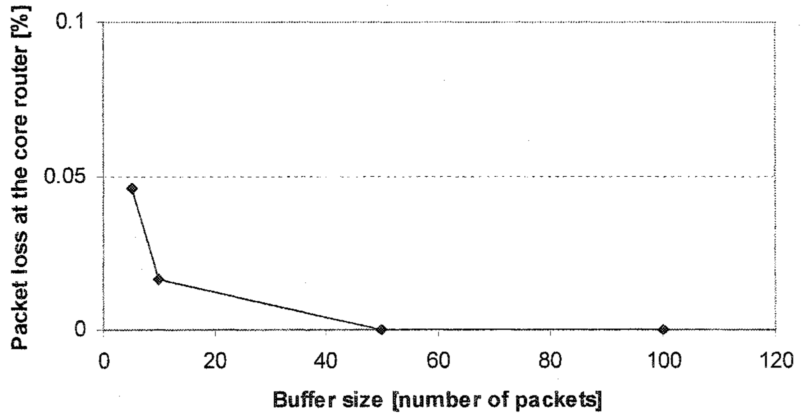

Figure 5-68a. The numerical value of packet loss [%] measured at the core router vs. B when α–stable traffic with α = 1.95 is used and the GE is used

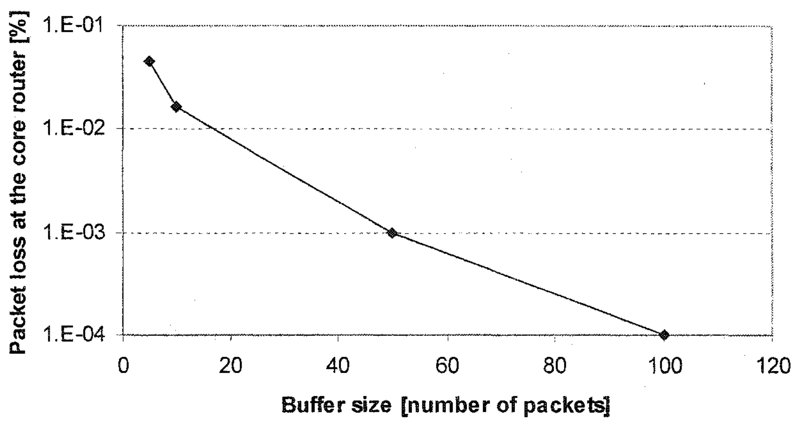

Figure 5-68b. The logarithm value of packet loss [%] measured at the core router vs. B when α–stable traffic with α = 1.95 is used and the GE is used

5.4 EVALUATION OF NON-GAUSSIAN α–STABLE ESTIMATOR (ASE)

Let us recall that the ASE operates as follows:

$$\hat{W}_j = m + K\sigma_W$$

where the parameter m is the mean of the number of arrivals per unit time, σ_W is the scale parameter. How to compute σ_W can be seen in [115]. The parameter K is determined by the value of overshoot probability ε and index of stability α. We repeat the same experiments described for ME and GE using ASE. Poisson and α–stable traffic with α = 1.95 and α = 1.60 have been used.

Experiment 5-9

We repeat Experiment 5-1 using ASE. We use the same parameters as indicated earlier to assess the effect of X on the performance. We set Y = 1 min and T_{meas} = 5 min. We apply buffer size of 10 packets and 100 packets at the core router. We run experiments using $\varepsilon = 0.01$ (refer to Table 3-2) for X: 5 seconds, 10 seconds, 15 seconds and 20 seconds.

Observe that the ASE shows similar behavior with the GE. The traffic processed with larger X is smoother than the process with smaller X, the mean and scale parameter calculated in the ASE formula become smaller with larger X, thus the allocation rate decreases, the packet loss, average delay and delay jitter increases. As far as the effect of the buffer size, when increasing the buffer size, the packet loss decreases and the delay increases. However, the IPDV jitter changes smoothly with the buffer size. In summary, all results appear being very close to those of GE.

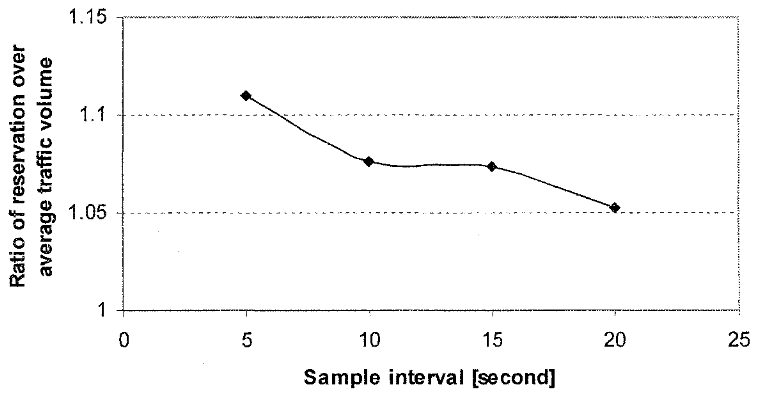

Figure 5-69. Ratio of reservation over average traffic volume when Poisson traffic is applied and the ASE is used

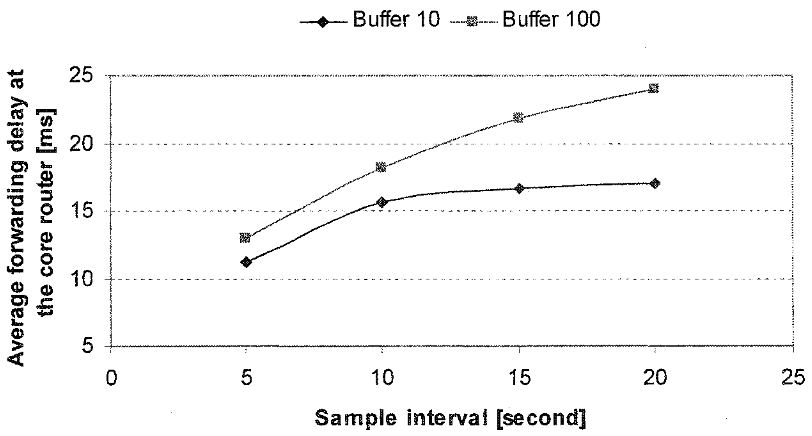

Figure 5-70. Average forwarding delay [ms] measured at the core router vs. X with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and the ASE is used

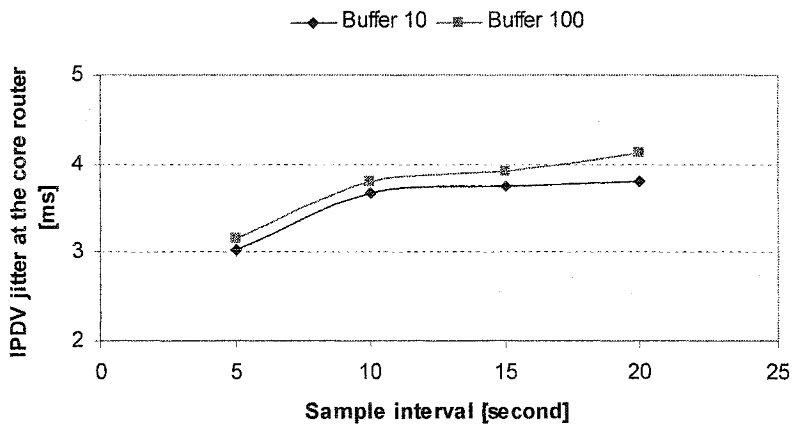

Figure 5-71. IPDV jitter [ms] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and the ASE is used

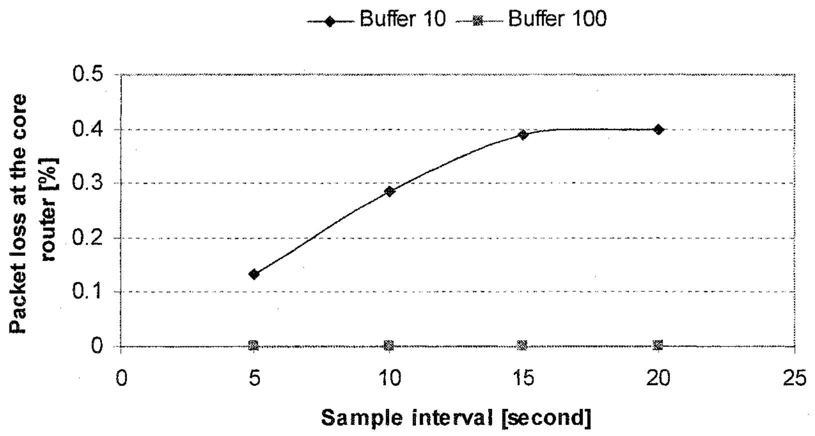

Figure 5-72a. The numerical value of packet loss [%] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and the ASE is used

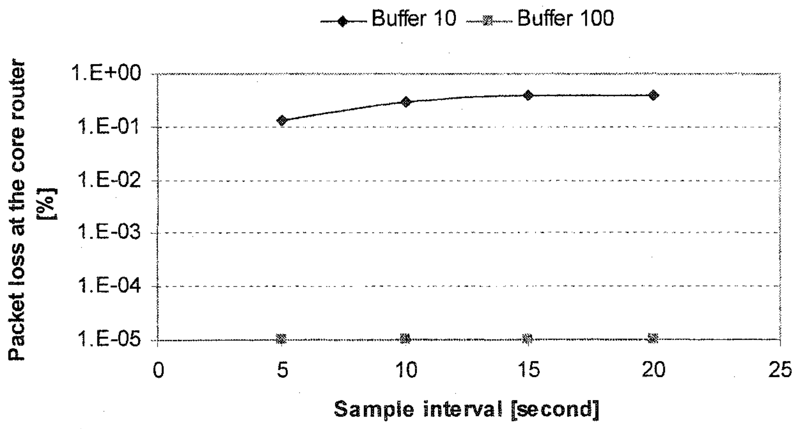

Figure 5-72b. The logarithm value of packet loss [%] measured at the core router vs. *X* with buffer size of a) 10 packets, b) 100 packets when Poisson traffic is applied and the ASE is used

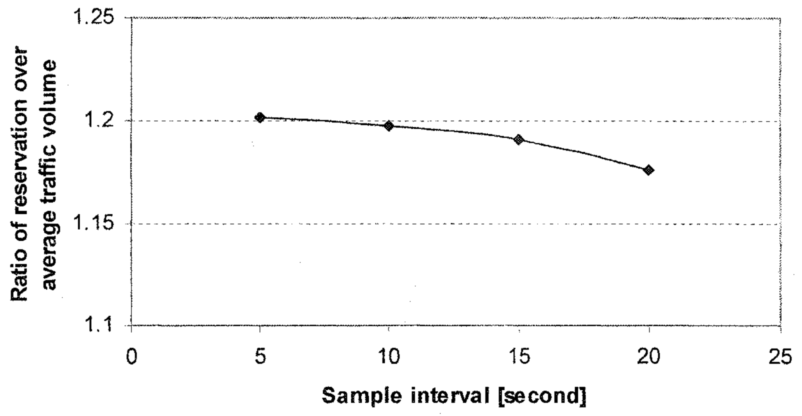

Figure 5-73. Ratio of reservation over average traffic volume when α–stable traffic with α = 1.95 is applied and the ASE is used

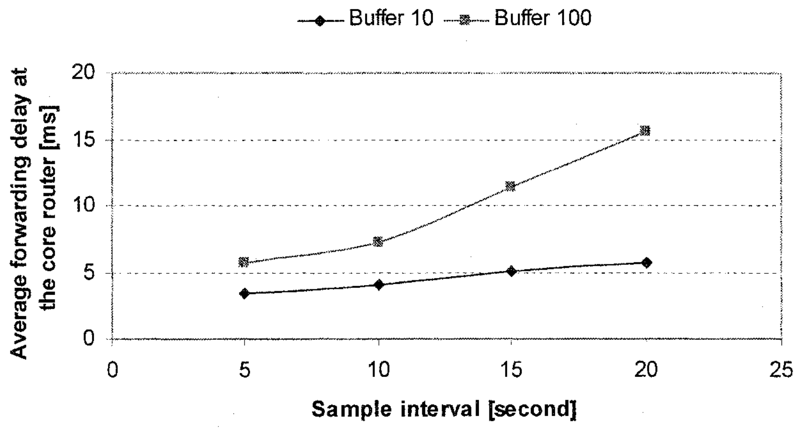

Figure 5-74. Average forwarding delay [ms] measured at the core router vs. X with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with α = 1.95 is applied and the ASE is used

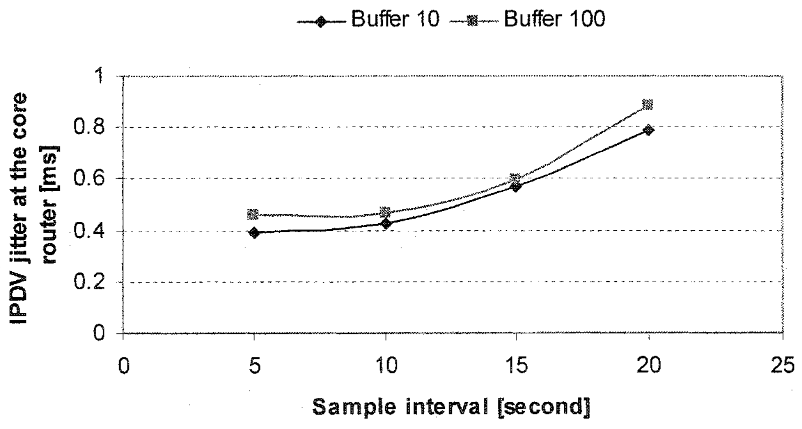

Figure 5-75. IPDV jitter [ms] measured at the core router vs. X with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with α = 1.95 is applied and the ASE is used

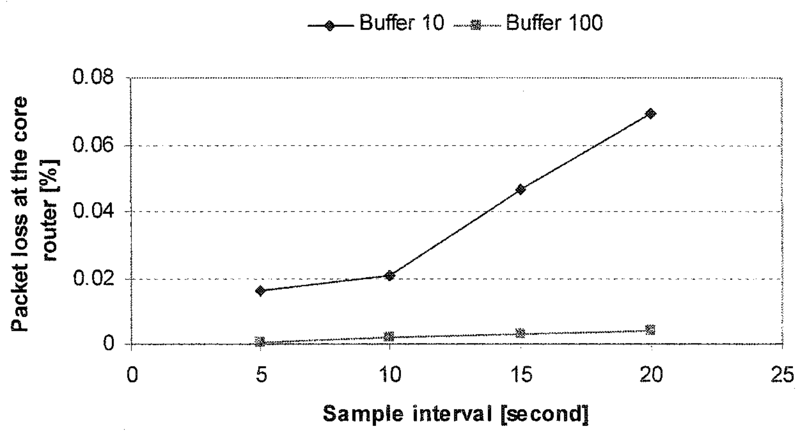

Figure 5-76a. The numerical value of packet loss [%] measured at the core router vs. X with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with $\alpha = 1.95$ is applied and the ASE is used

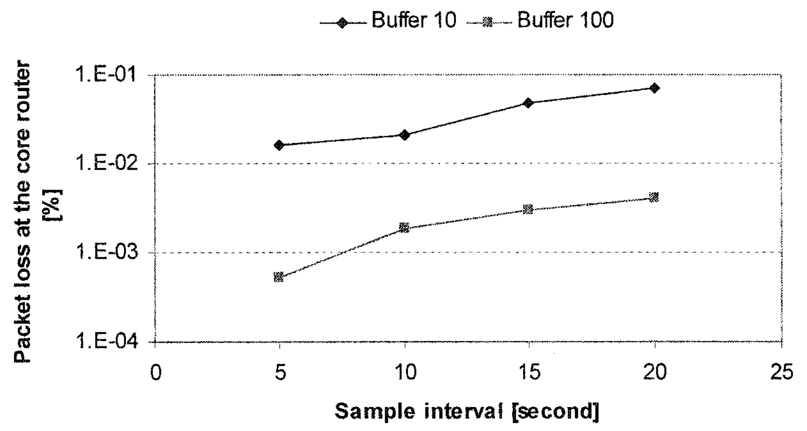

Figure 5-76b. The logarithm value of packet loss [%] measured at the core router vs. X with buffer size of a) 10 packets, b) 100 packets when α–stable traffic with $\alpha = 1.95$ is applied and the ASE is used

Experiment 5-10

We repeat Experiment 5-2 but now using the ASE. We use the same parameters as indicated earlier to assess the effect of T_{meas} and margin size on the performance. We set X = 5 seconds, Y = 1 min, buffer size 10 packets. We start running for T_{meas} = W*X with W = 12, 24, 36, 60 (T_{meas} = 1 min, 2 min, 3 min and 5 min). The margin is controlled by ε. In order to keep the comparability with GE, the ε value is chosen 0.1, 0.01 and 0.001 (corresponding to k = 1, 2 and 3 for the GE case).

ASE displays similar behavior with GE and ME. Larger T_{meas} provides more samples, it will have more chance to get higher variance of samples resulting in higher allocation rate. Observe that when the T_{meas} increases, the allocation rate increases, thus the average forwarding delay, IPDV jitter and packet loss reduces. When comparing the performance for $\alpha = 1.95$ and $\alpha = 1.60$, we realize that for $\alpha = 1.60$ we have higher allocation rates. The difference in terms of allocation between $\alpha = 1.95$ and $\alpha = 1.60$ case increases in favor of $\alpha = 1.60$ (burstiner traffic) as the factor ε becomes smaller. As can be seen from Table 3-2 and 3-3, the value of K produced for the same ε value, increase faster as ε becomes smaller for the burstiner traffic. This forces the ASE to allocate higher bandwidth. As consequence, the performance of the $\alpha = 1.60$ stream is better as compared to $\alpha = 1.95$.

Comparing with GE and ME, the results for Poisson traffic are quite close for GE and ME.

ASE seems to be able to make better assessment of the needs of the traffic when coming to α–stable traffic. The amount of resources requested is higher for both $\alpha = 1.95$ and $\alpha = 1.60$, especially for the cases $\varepsilon = 0.01$, $\varepsilon = 0.001$, which are actually more of interest, since $\varepsilon = 0.1$ (and k=1) produce high packet loss rate.

ASE seems to be able to produce less losses, lower average delays. This advantage over GE becomes more evident for $\alpha = 1.60$, indicating that it is more capable of "tracking" the intensive burstiness associated with α–stable processes as compared to the GE.

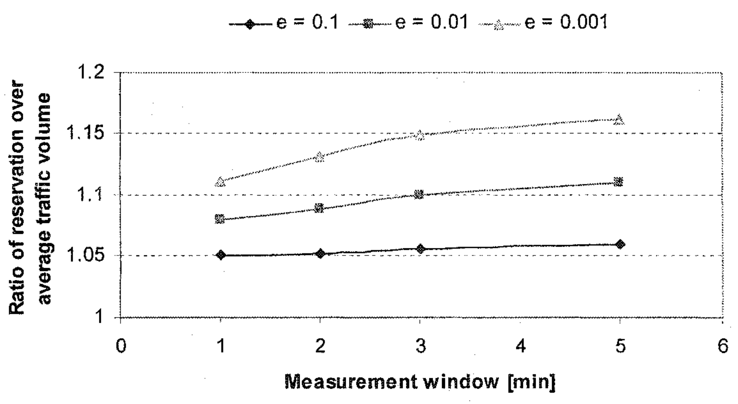

Figure 5-77. Ratio of reservation over average traffic volume when Poisson traffic is applied and the ASE is used

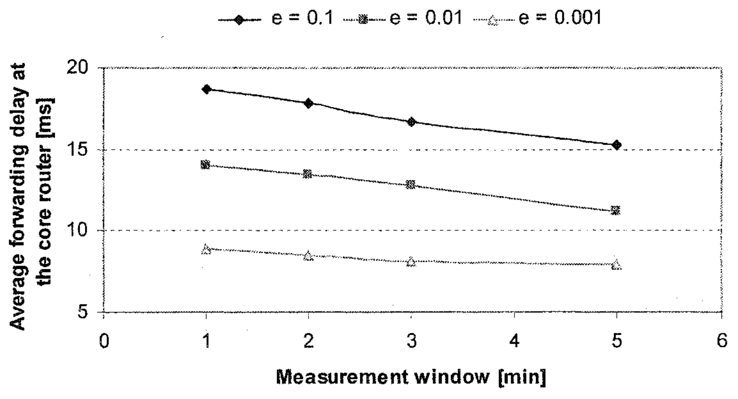

Figure 5-78. Average forwarding delay [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the ASE is used

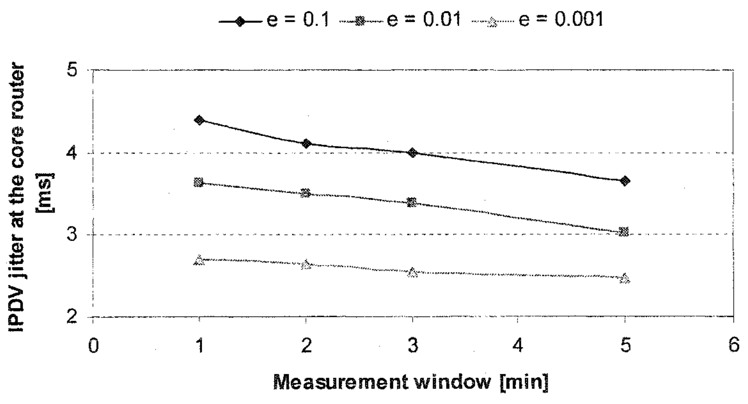

Figure 5-79. IPDV jitter [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the ASE is used

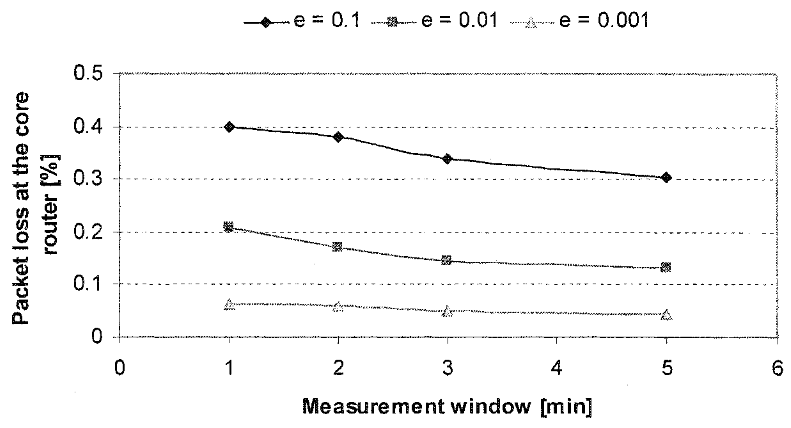

Figure 5-80a. The numerical value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the ASE is used

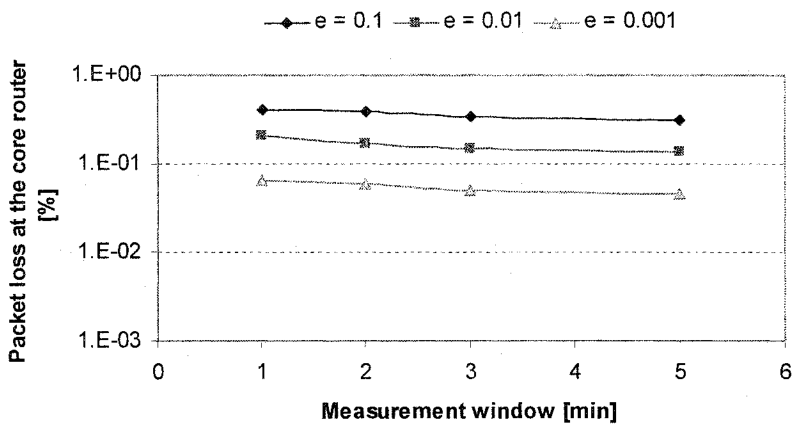

Figure 5-80b. The logarithm value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when Poisson traffic is applied and the ASE is used

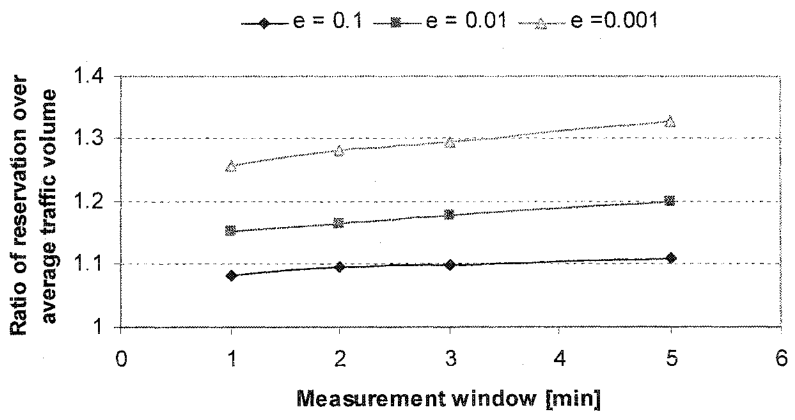

Figure 5-81. Ratio of reservation over average traffic volume vs. T_{meas} with buffer size of 10 packets when α-stable traffic with $\alpha = 1.95$ is applied and the ASE is used

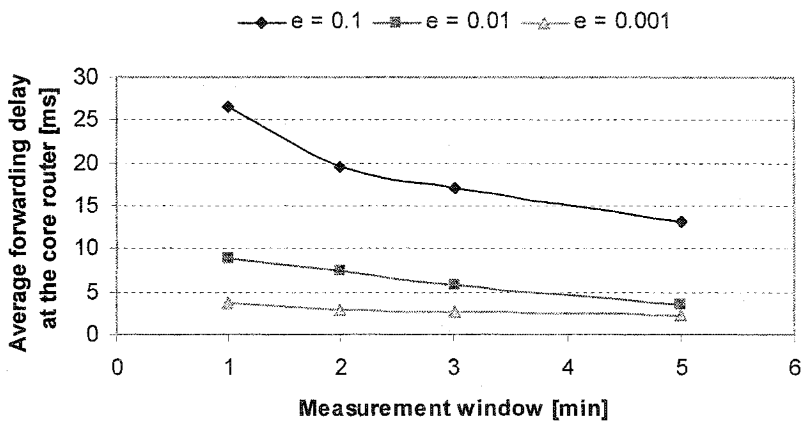

Figure 5-82. Average forwarding delay [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with $\alpha = 1.95$ is applied and the ASE is used

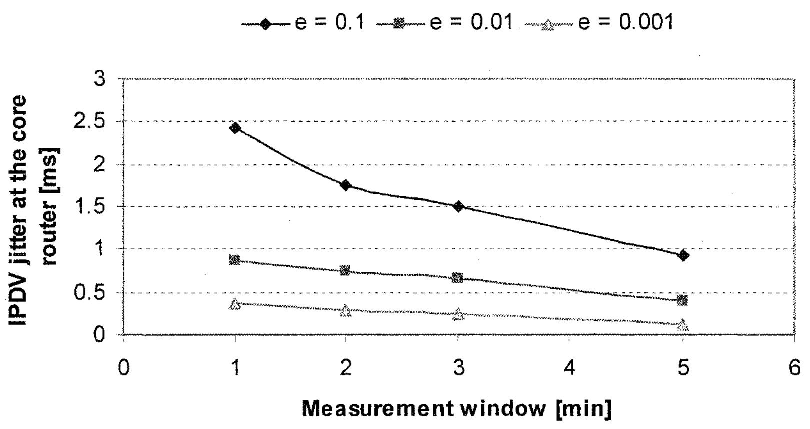

Figure 5-83. IPDV jitter [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with $\alpha = 1.95$ is applied and the ASE is used

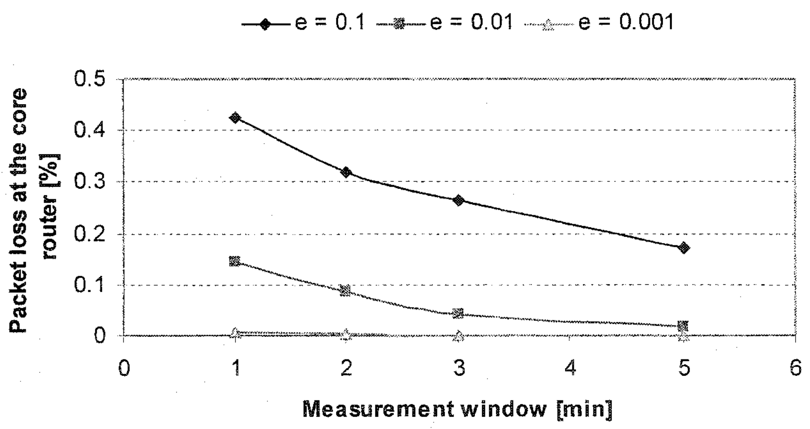

Figure 5-84a. The numerical value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with $\alpha = 1.95$ is applied and the ASE is used

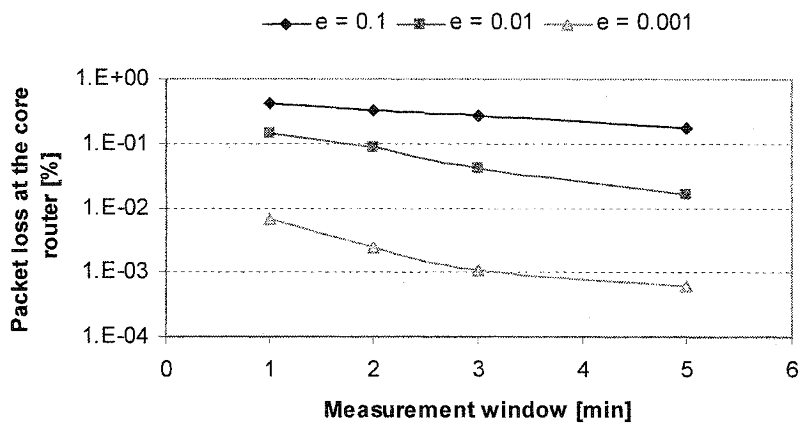

Figure 5-84b. The logarithm value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with α = 1.95 is applied and the ASE is used

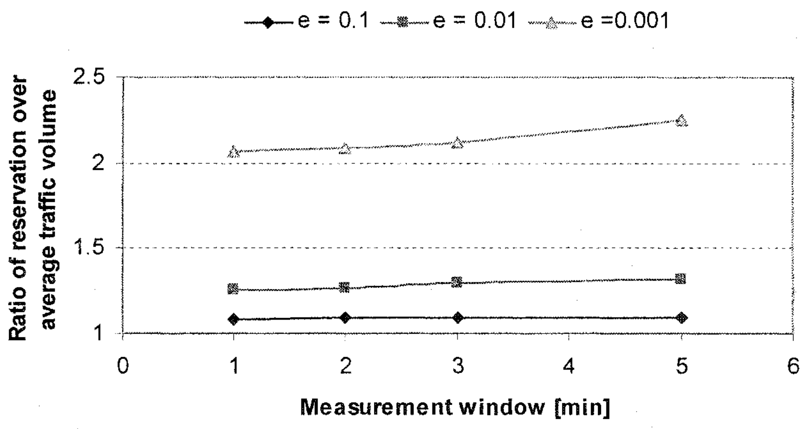

Figure 5-85. Ratio of reservation over average traffic volume vs. T_{meas} with buffer size of 10 packets when α-stable traffic with α = 1.60 is applied and the ASE is used

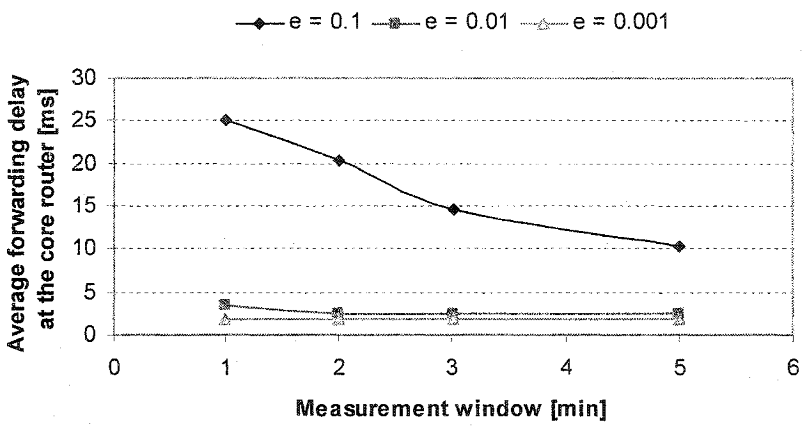

Figure 5-86. Average forwarding delay [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with α = 1.60 is applied and the ASE is used

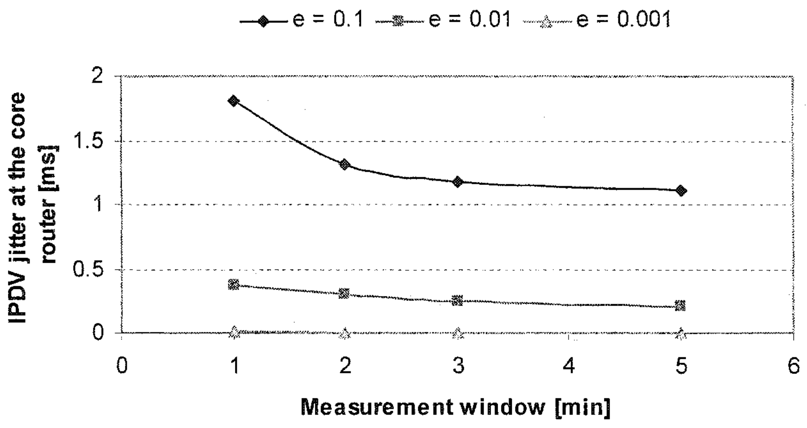

Figure 5-87. IPDV jitter [ms] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with α = 1.60 is applied and the ASE is used

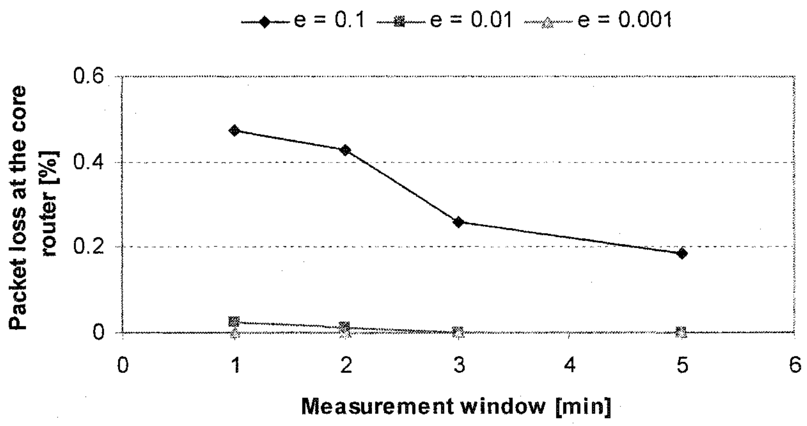

Figure 5-88a. The numerical value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with α = 1.60 is applied and the ASE is used

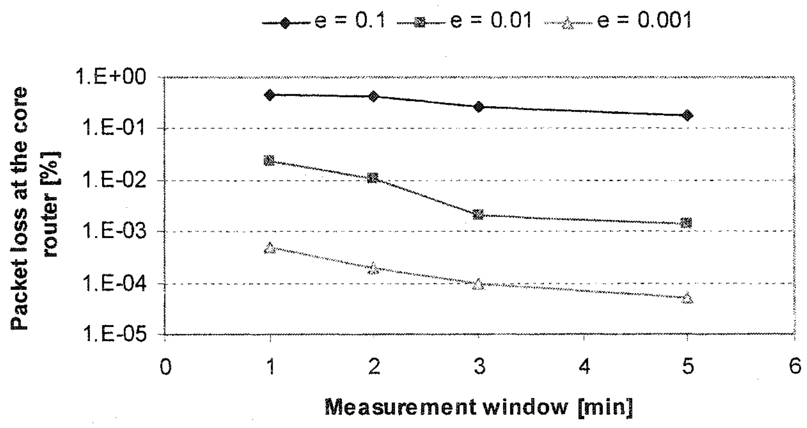

Figure 5-88b. The logarithm value of packet loss [%] measured at the core router vs. T_{meas} with buffer size of 10 packets when α-stable traffic with α = 1.60 is applied and the ASE is used

Experiment 5-11

We repeat Experiment 5-3 with ASE to assess how the performance changes as Y (Reservation window) changes. We set $X = 5$ second, $T_{meas} = 5$ min, buffer size 10 packets, and we run experiments with different values of Y ($Y = 1$ min, 2 min, 3 min and 5 min). We choose margin $\varepsilon = 0.01$ and plot the various performance curves versus Y.

ASE shows similar behavior with GE and ME. The system performance doesn't change much for different value of Y.

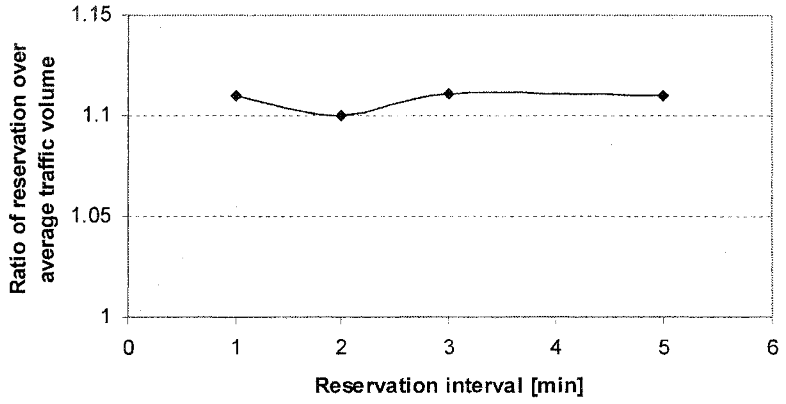

Figure 5-89. Ratio of reservation over average traffic volume when Poisson traffic is applied and the ASE is used

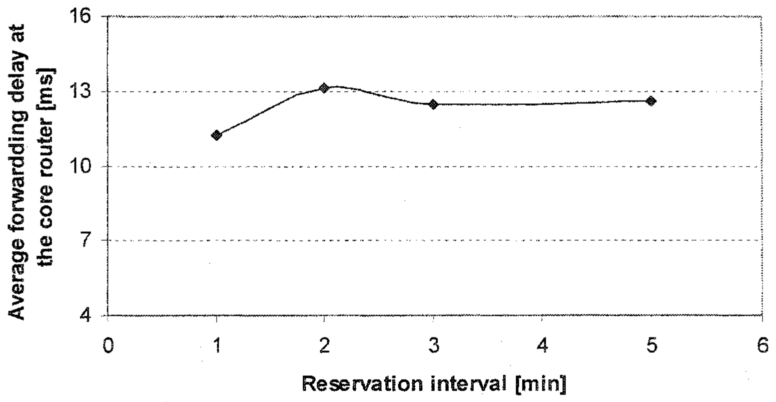

Figure 5-90. Average forwarding delay [ms] measured at the core router vs. Y with buffer size of 10 packets when Poisson traffic is applied and the ASE is used

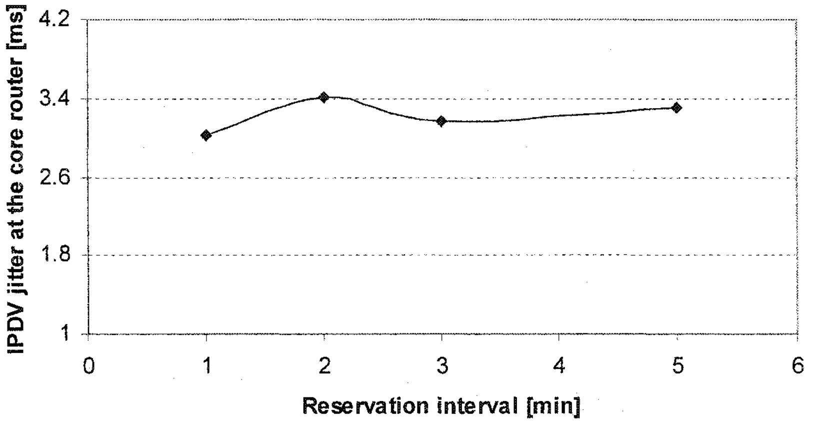

Figure 5-91. IPDV jitter [ms] measured at the core router vs. *Y* with buffer size of 10 packets when Poisson traffic is applied and the ASE is used

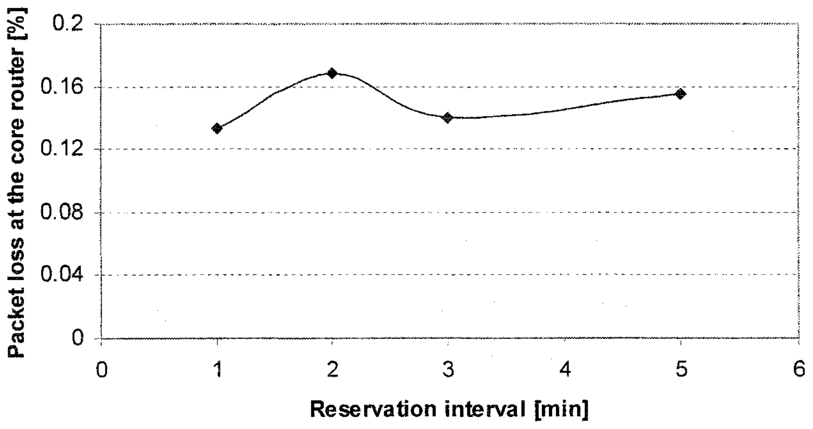

Figure 5-92a. The numerical value of packet loss [%] measured at the core router vs. *Y* with buffer size of 10 packets when Poisson traffic is applied and the ASE is used

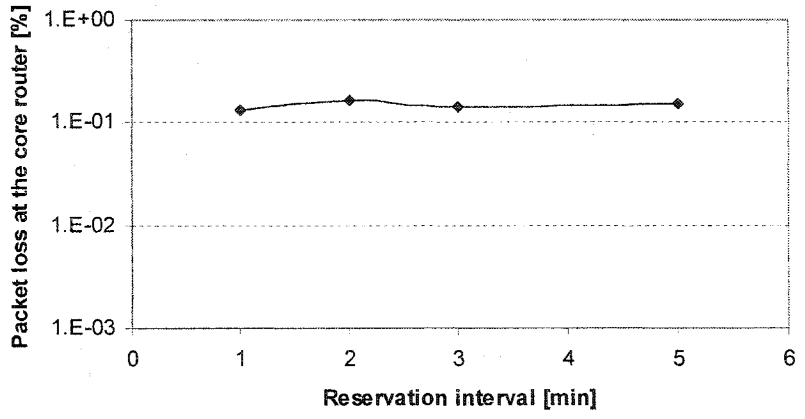

Figure 5-92b. The logarithm value of packet loss [%] measured at the core router vs. *Y* with buffer size of 10 packets when Poisson traffic is applied and the ASE is used

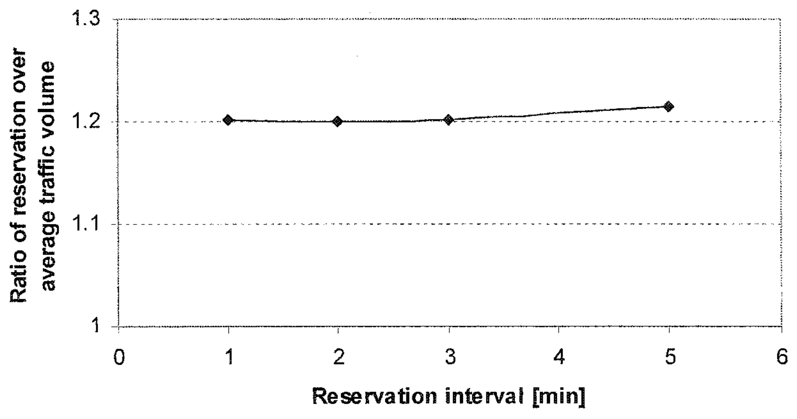

Figure 5-93.Ratio of reservation over average traffic volume when α–stable traffic with α = 1.95 is applied and the ASE is used

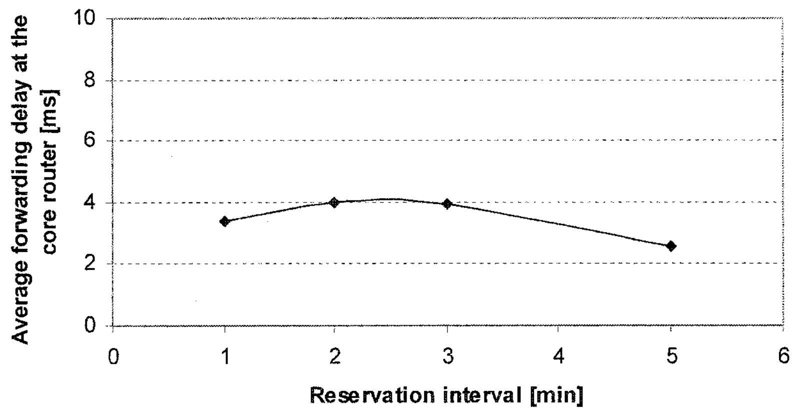

Figure 5-94. Average forwarding delay [ms] measured at the core router vs. *Y* with buffer size of 10 packets when α–stable traffic with α = 1.95 is applied and the ASE is used

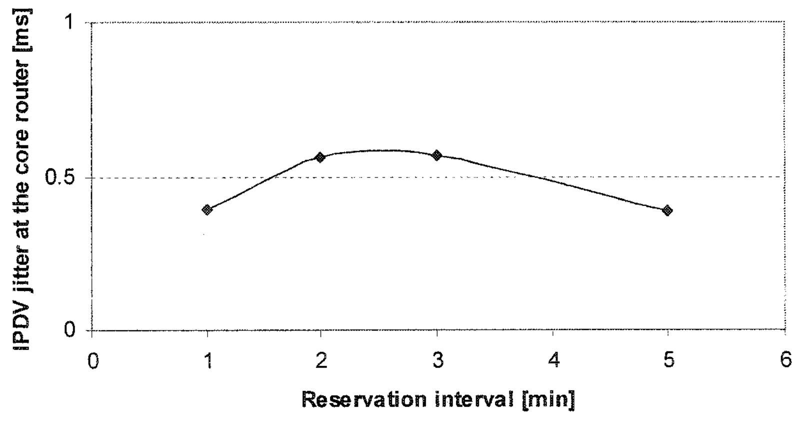

Figure 5-95. IPDV jitter [ms] measured at the core router vs. *Y* with buffer size of 10 packets when α–stable traffic with α = 1.95 is applied and the ASE is used

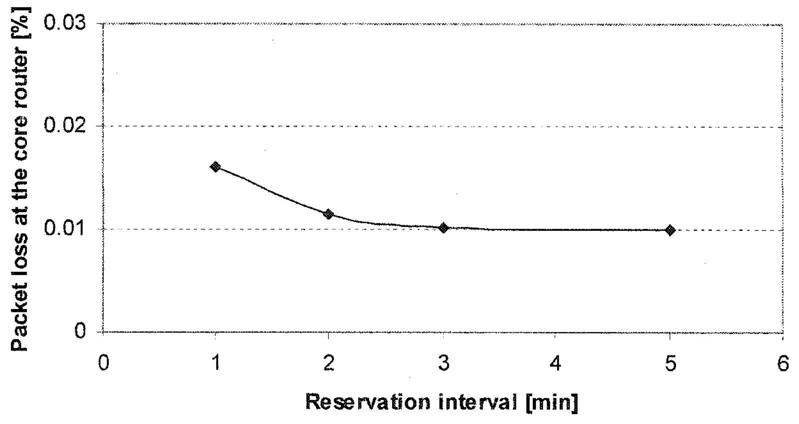

Figure 5-96a. The numerical value of packet loss [%] measured at the core router vs. *Y* with buffer size of 10 packets when α–stable traffic with α = 1.95 is applied and the ASE is used

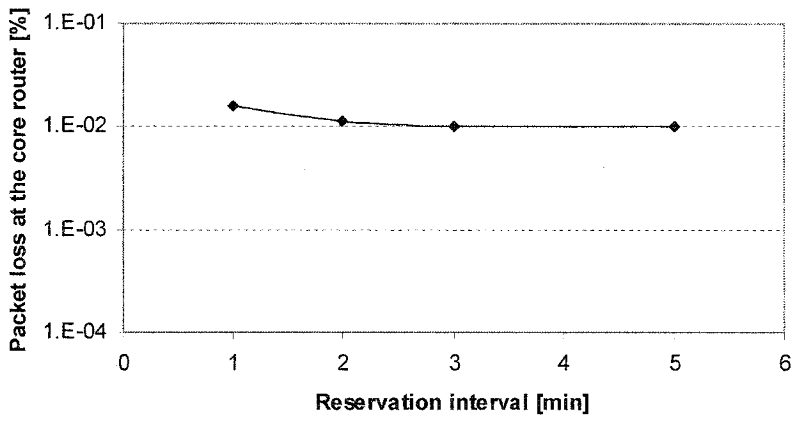

Figure 5-96b. The logarithm value of packet loss [%] measured at the core router vs. *Y* with buffer size of 10 packets when α–stable traffic with α = 1.95 is applied and the ASE is used

Experiment 5-12

We repeat Experiment 5-4 using the ASE to assess the effect of buffer size B on the performance. We choose a value of $X = 5$ seconds, $T_{meas} = 5$ min and $Y = 1$ min. We run the estimator with margin $\varepsilon = 0.01$. The buffer size is chosen from 5 packets, 10 packets, 50 packets and 100 packets. The results are similar to the case of GE, with only some lower levels of average delay appearing for α–stable traffic.

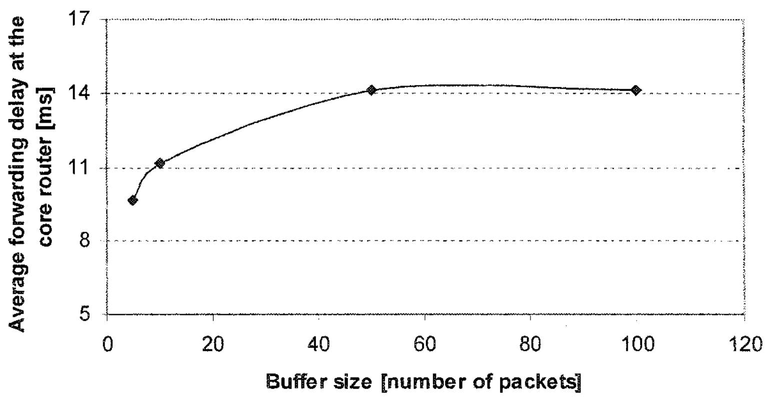

Figure 5-97. Average forwarding delay [ms] measured at the core router vs. B when Poisson traffic is applied and the ASE is used

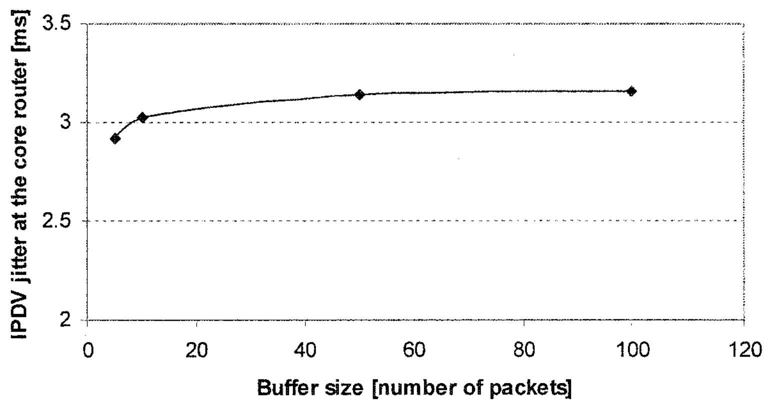

Figure 5-98. IPDV jitter [ms] measured at the core router vs. B when Poisson traffic is applied and the ASE is used

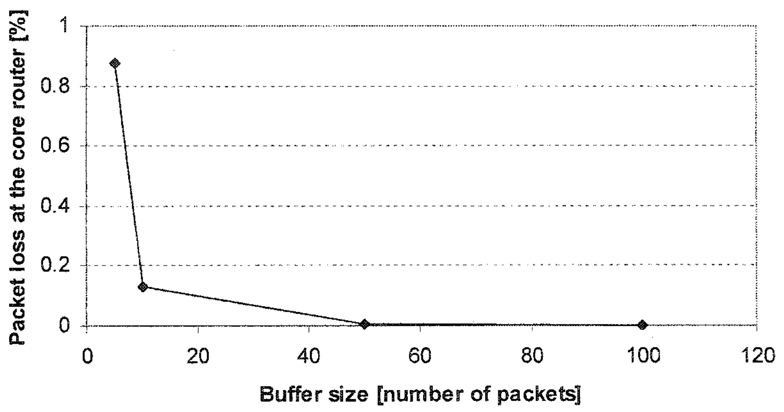

Figure 5-99a. The numerical value of packet loss [%] measured at the core router vs. B when Poisson traffic is applied and the ASE is used

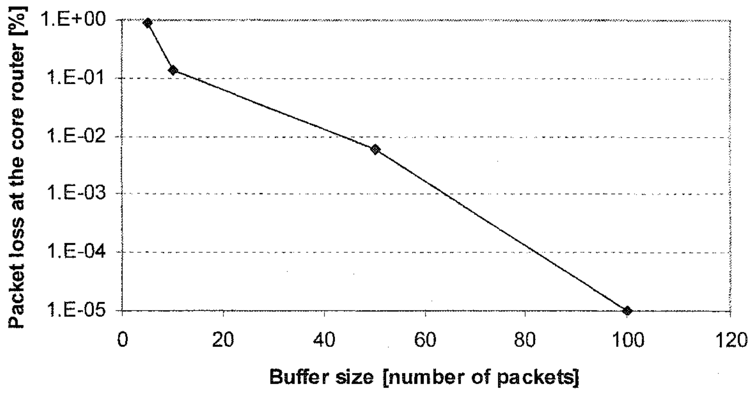

Figure 5-99b. The logarithm value of packet loss [%] measured at the core router vs. *B* when Poisson traffic is applied and the ASE is used

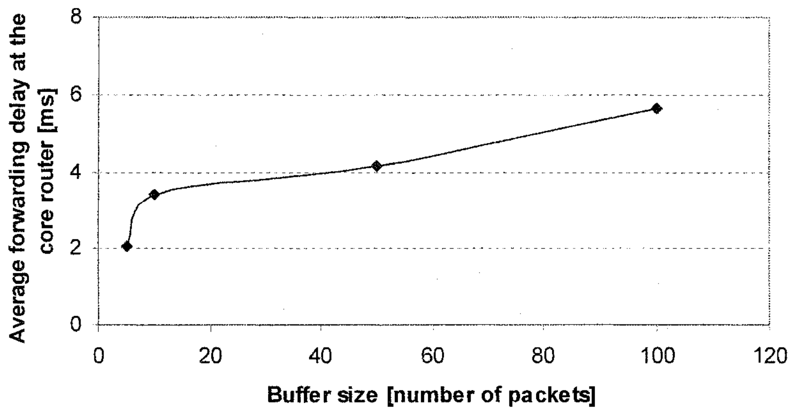

Figure 5-100. Average forwarding delay [ms] measured at the core router vs. *B* when α–stable traffic with α = 1.95 is applied and the ASE is used

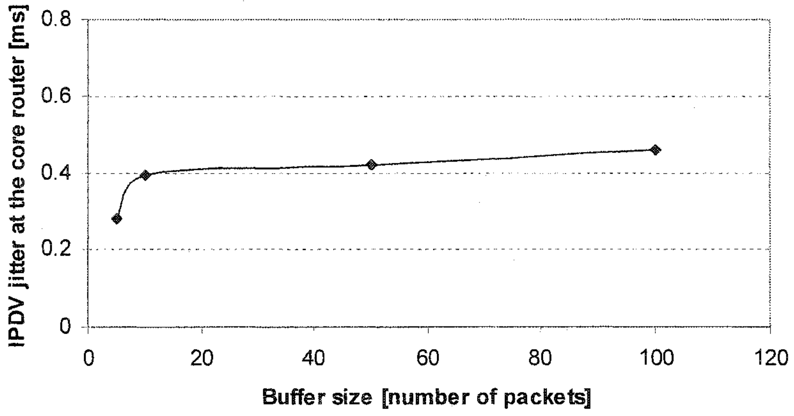

Figure 5-101. IPDV jitter [ms] measured at the core router vs. *B* when α–stable traffic with α = 1.95 is applied and the ASE is used

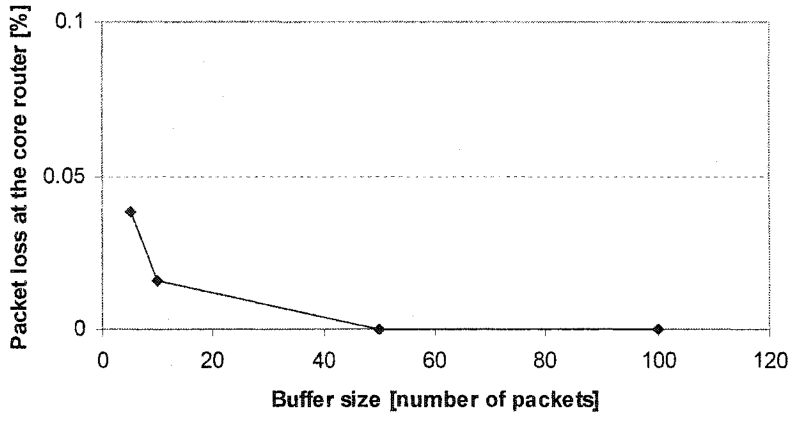

Figure 5-102a. The numerical value of packet loss [%] measured at the core router vs. B when α–stable traffic with $\alpha = 1.95$ is applied and the ASE is used

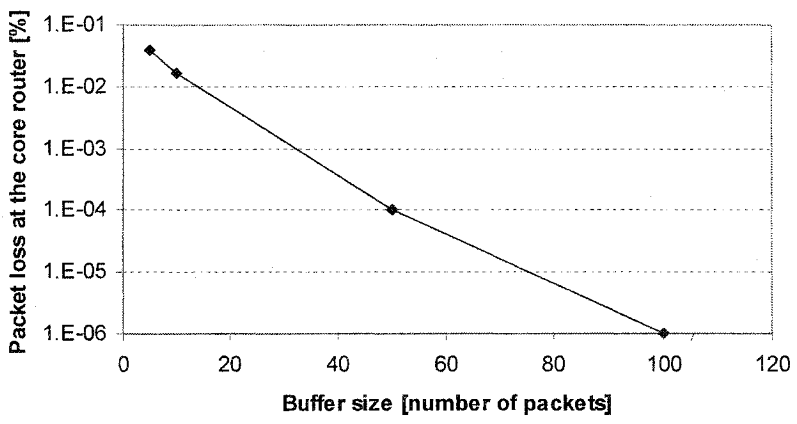

Figure 5-102b. The logarithm value of packet loss [%] measured at the core router vs. B when α–stable traffic with $\alpha = 1.95$ is applied and the ASE is used

5.5 SUMMARY OF ESTIMATORS

We already have measured the effect of X, T_{meas}, Y and B on the system's performance for the three estimators. In order to compare the GE and ASE (both of them belong to the envelope process based estimator), we extract them together. Here we only show the performance as function of T_{meas} when α–stable traffic is used. We consider the case the traffic process exceeds its envelope by: 0.1, 0.01 and 0.001. The corresponding values of the parameter K for ASE and parameter a for GE required to achieve these levels of the envelope exceeding the set threshold are obtained from the Table 3-1, 3-2 and 3-3. We show the values of K and a in Table 5-1.

Observe that GE and ASE show similar performance when high levels of losses are tolerated. The difference among each other becomes more noticeable when the QoS requirements become more stringent. This is due to fact that the required values for parameter K and a are of similar magnitude for both processes when high levels of losses are acceptable (see Table 5-1). The values of these parameters have to increase in order to achieve smaller levels of losses. However, due to the different decay rate of the loss probability as function of K and a, significantly larger values are required in the α–stable than the Gaussian case. Thus the α–stable distribution produces higher allocation rates as illustrated in Figures 5-103 ($\alpha = 1.95$) and 5-107 ($\alpha = 1.60$), thus achieves better performance as shown in Figures 5-104, 5-105, 5-106, 5-108, 5-109 and 5-110.

Please note that differently from the Gaussian case, the α–stable estimator is able to provide better performance when QoS requirement becomes more stringent.

From these tests it is concluded that the dynamic bandwidth allocation algorithm based on the non-Gaussian α–stable envelope process (ASE) results in a more reliable way to provide quality of service guarantee when the traffic exhibits high variability.

Based on our extensive experiments, some conclusions can be drawn:

- Maximum Estimator (ME) is the weakest of all estimators. It is simple but wasteful of resources.

- Gaussian Estimator (GE) assumes that the incoming aggregate traffic follows Gaussian distribution.

- Non-Gaussian α–stable Estimator (ASE) will be appropriate for aggregate traffic patterns with very high degree of burstiness.

Another conclusion we can draw by analyzing the system parameters that are found to have a significant impact on performance:

- Sample interval X: By increasing X the allocation rate decreases, the delay, jitter and packet loss increase. So small X is better as far as it does not generate to the router excessive computation load.

- Measurement window T_{meas}: T_{meas} plays a key role to system's performance. Larger T_{meas} achieves better performance. However T_{meas} controls the trade-off between over-provisioning and QoS performance. In addition, it controls the speed by which the system reacts to a change in the average volume of transported information. A value of T_{meas} in the range of 2 minutes seems to be a good choice.

- Resource reservation window Y: When Y increases, system performance does not change very much. However, smaller Y means more frequently sending control traffic to the nodes on the path, leading to greater signaling overhead. At the same time, bigger value of Y reduces the speed by which the system would react to changing traffic conditions. These trade-offs should be taken into consideration when choosing the value of Y.

- Buffer size at the router(s) B: By increasing the buffer size, the packet loss decreases, but the delay increases. Buffer size controls the trade-off between losses and delay performance.

Table 5-1. Value of the parameter K for ASE and parameter a for GE corresponding to different values of the violation of the set threshold

Estimators \ Probability	0.1	0.01	0.001
GE	$a = 1.2815$	$a = 2.3263$	$a = 3.0902$
ASE ($\alpha = 1.95$)	$K = 1.82$	$K = 3.45$	$K = 5.83$
ASE ($\alpha = 1.60$)	$K = 1.98$	$K = 6$	$K = 23.93$

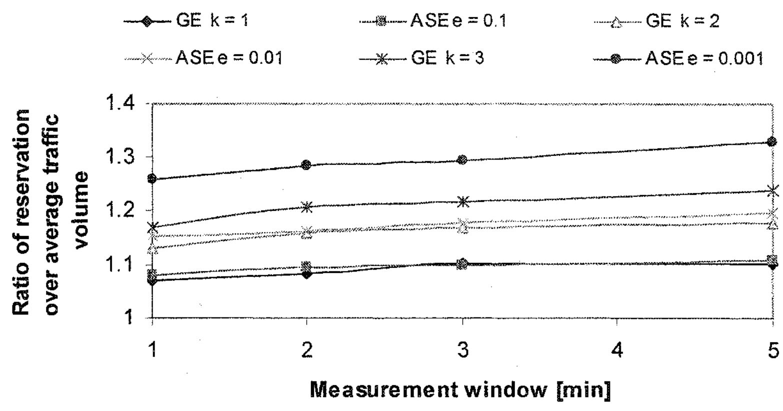

Figure 5-103. Ratio of reservation over average traffic volume vs. T_{meas} for GE and ASE when α–stable traffic with α = 1.95 is applied

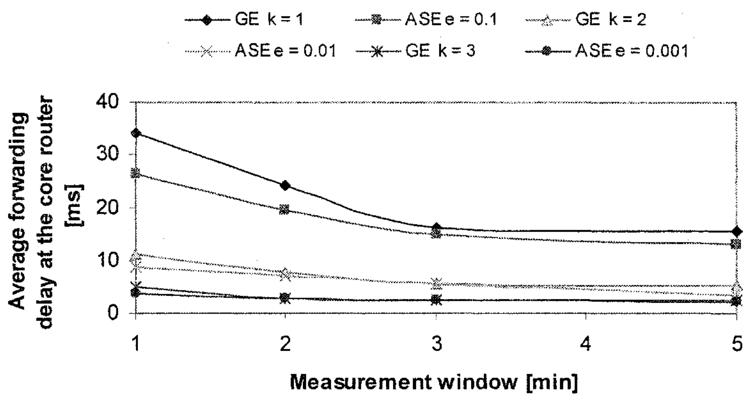

Figure 5-104. Average forwarding delay [ms] measured at the core router vs. T_{meas} for GE and ASE when α–stable traffic with α = 1.95 is applied

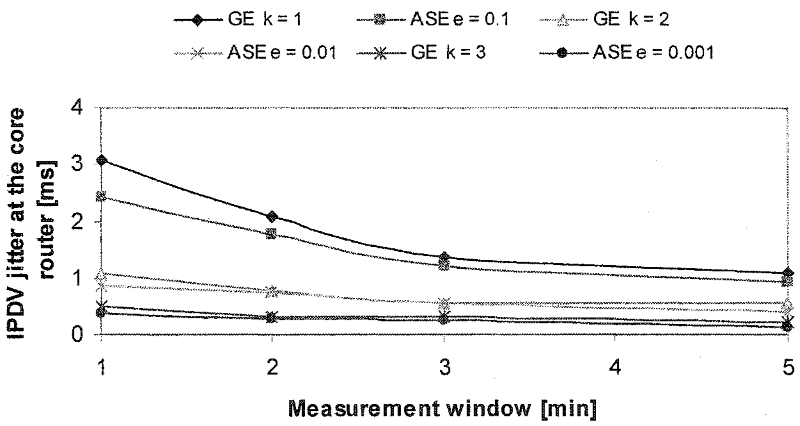

Figure 5-105. IPDV jitter [ms] measured at the core router vs. T_{meas} for GE and ASE when α–stable traffic with α = 1.95 is applied

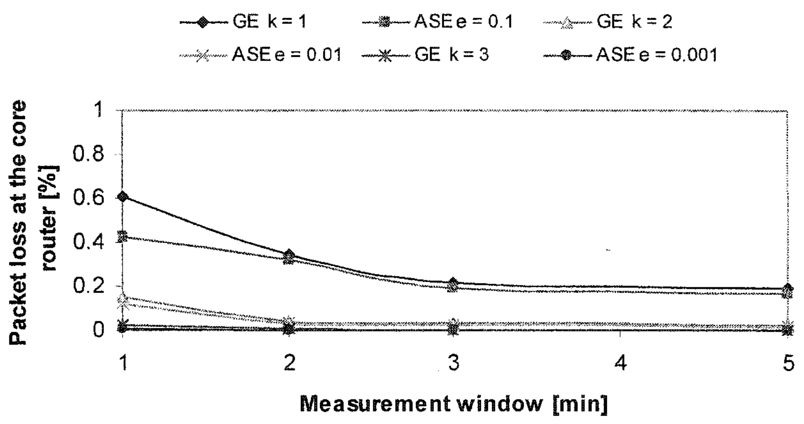

Figure 5-106a. The numerical value of packet loss [%] measured at the core router vs. T_{meas} for GE and ASE when α–stable traffic with α = 1.95 is applied

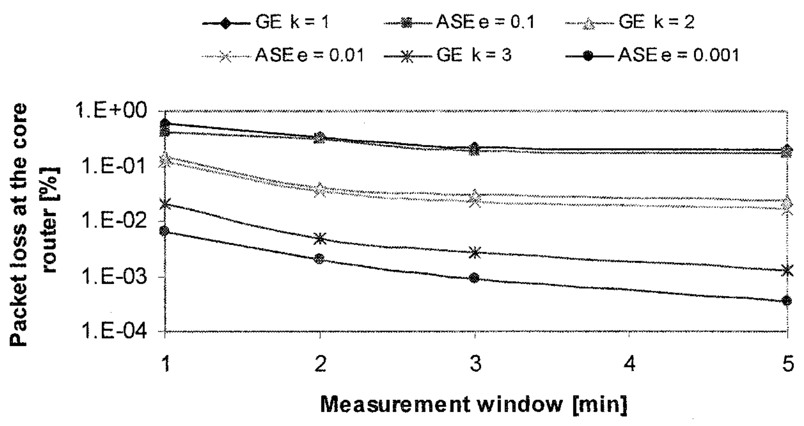

Figure 5-106b. The logarithm value of packet loss [%] measured at the core router vs. T_{meas} for GE and ASE when α–stable traffic with α = 1.95 is applied

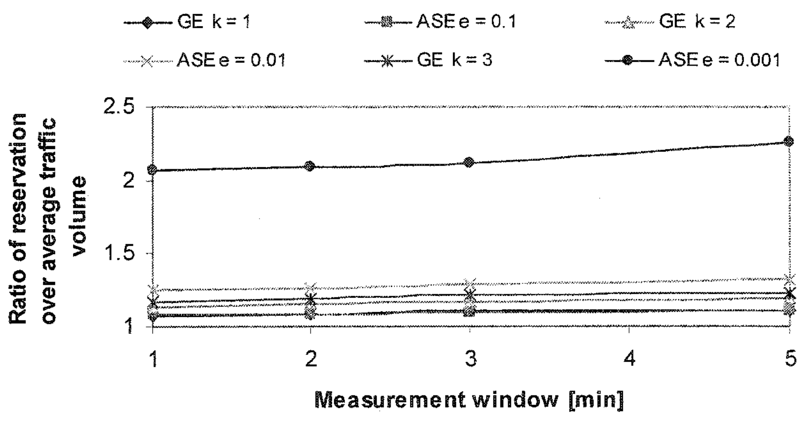

Figure 5-107. Ratio of Ratio of reservation over average traffic volume vs. T_{meas} for GE and ASE when α–stable traffic with α = 1.60 is applied

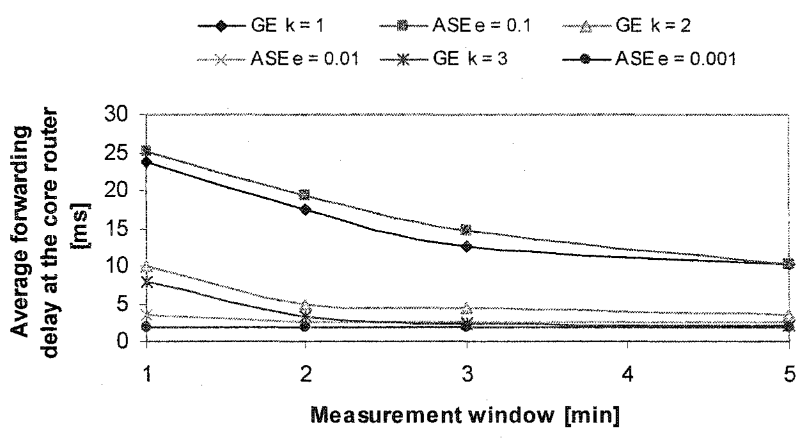

Figure 5-108. Average forwarding delay [ms] measured at the core router vs. T_{meas} for GE and ASE when α–stable traffic with α = 1.60 is applied

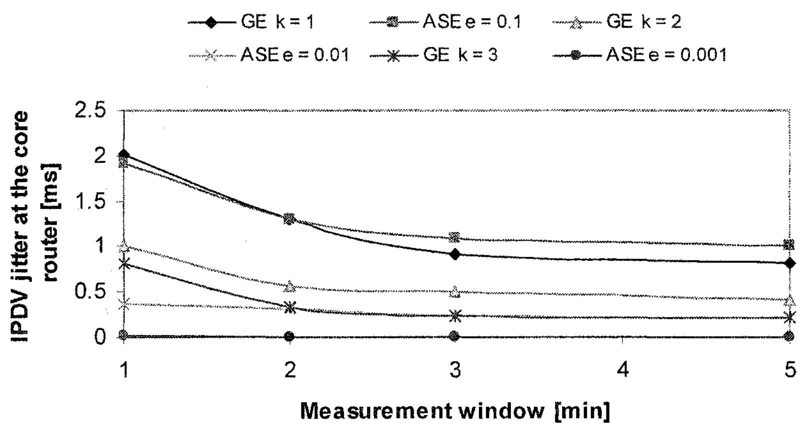

Figure 5-109. IPDV jitter [ms] measured at the core router vs. T_{meas} for GE and ASE when α–stable traffic with α = 1.60 is applied

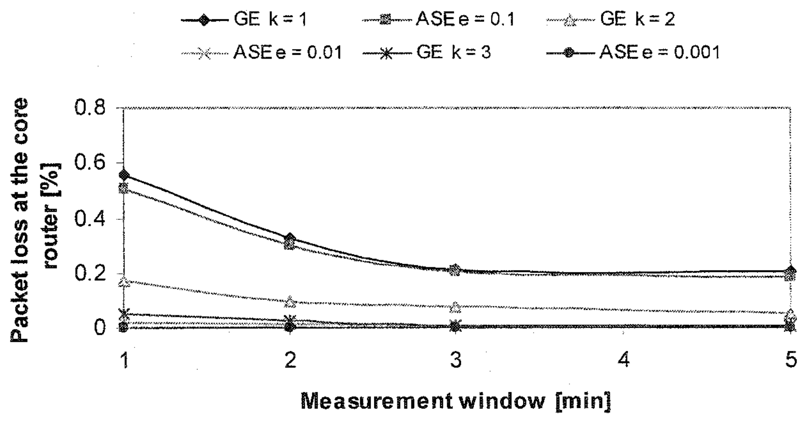

Figure 5-110a. The numerical value of packet loss [%] measured at the core router vs. T_{meas} for GE and ASE when α–stable traffic with α = 1.60 is applied

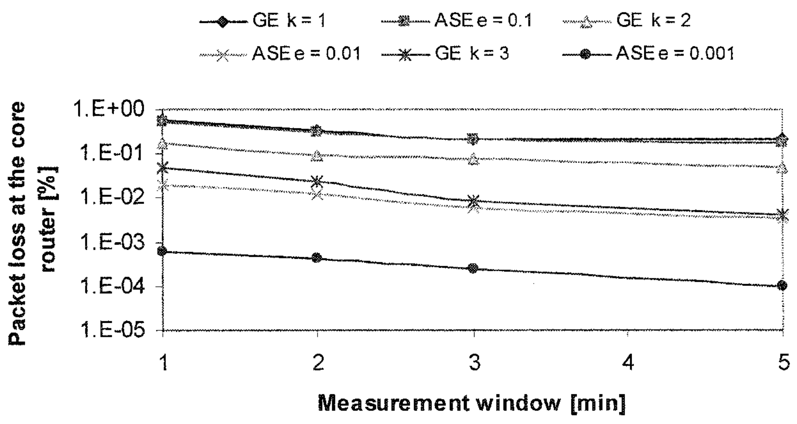

Figure 5-110b. The logarithm value of packet loss [%] measured at the core router vs. T_{meas} for GE and ASE when α–stable traffic with $\alpha = 1.60$ is applied

Chapter 6

CONCLUSION AND FUTURE WORKS

This thesis presents a framework for providing dynamic Quality of Service (QoS) support in Virtual Private Networks (VPNs). Specifically, we focus on MPLS based VPN, with the QoS components including use of MPLS Diffserv, MPLS Traffic Engineering, RSVP-TE signaling protocol etc.

The Dynamic Bandwidth Allocation (DBA) model focuses on dynamic resource allocation using various forms of traffic estimators. Three estimation algorithms are used. The Dynamic Classed Based Queuing (CBQ) technique is applied for the provision and policing of resources. The implemented system can automatically adjust the bandwidth size of a VPN tunnel based on how much traffic is flowing through the tunnel. An ISP can simplify the task of managing its network and reduces costs by using our DBA mechanism, taking advantage of available tunnel bandwidth while still providing guarantees for high-priority traffic.

We implemented and evaluated the DBA model on our MPLS Diffserv enabled Linux test-bed. Test-bed is one more effective way to test the system performance compared with simulations. Our test-bed consists of two customer networks and one MPLS backbone network. The MPLS backbone network is formed by two edge routers and one core router, which establish one Diffserv enabled LSP. All routers run under the Linux RedHat^{TM} system. The kernel version for our application is 2.4.19. In our implementation, the RSVP-TE daemon running under each MPLS router is responsible for the RSVP-TE signaling and the maintenance of the MPLS state. We carried out performance analysis by using different types of traffic.

Future work, extending this research, can be the provision of end-to-end QoS among multiple MPLS Diffserv domains and the way QoS characteristics are propagated across multiple domains. This is particularly interesting when integrating the current test-bed with a wireless domain or an optical network running GMPLS.

REFERENCES

[1] R. Comerford, "State of the Internet: Roundtable 4.0", IEEE Spectrum, Oct. 1998.

[2] P. Ferguson and G. Huston, "Quality of Service", John Wiley & Sons, 1998

[3] X. Xiao and L. Ni, "Internet QoS: A Big Picture", IEEE Network Magazine, March/April, pp. 8-18, 1999.

[4] B. Gleeson, A. Lin. J. Heinanen, G. Armitage. RFC2764 "A Framework for IP Based Virtual Private Networks" Feb. 2000

[5] S. Fotedar, M.Gerla, P. Crocetti and L. Fratta, "ATM Virtual Private Networks, " Communications of the ACM, vol.38, pp. 101-109, Feb 1995

[6] K. Muthukrishnan and A. Malis, "Core IP VPN Architecture." Draft-muthukrishnan-corevpn-arch-00.txt, October 1998

[7] D. Jamieson, B. Jamoussi, G. Wright and P. Beaubien, "MPLS VPN Architecture." Draft-jamieson-mpls-vpn-00.txt, August 1998

[8] E. Rosen, Y. Rekhter. RFC2547 "BGP/MPLS VPNs" March 1999

[9] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", Internet RFC 1633, Jun.1994.

[10] S. Shenker, C. Partridge and R. Guerin, " Specification of Guaranteed Quality of Service", RFC 2212, Sept.1997

[11] J. Wroclawski, "Specification of the Controlled-Load Network Element Service", RFC 2211, Sept.1997

[12] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource Reservation Protocol (RSVP) – Version 1 Functional Specification", RFC 2205, Sept. 1997.

[13] A. Mankin, F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, L. Zhang, "Resource ReSerVation Protocol (RSVP) Version 1 - Applicability Statement - Some Guidelines on Deployment", RFC 2208, 1997

[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC 2475, Dec.1998.

[15] H. Zhang, "Service Discipline For Guaranteed Performance Service in Packet Switching Networks", Proceedings of the IEEE, 83(10), Oct. 1995.

[16] J. Postel, "Service Mapping", RFC 795, Sept.1981

[17] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the Ipv4 and Ipv6 Headers", RFC 2474, 1998

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

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

[20] Chuck Semeria, "Supporting Differentiated Service Classes: Queue Scheduling Disciplines", Juniper Networks. December 2001

[21] Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, August 1993

[22] P.F.Chimento, "Standard Token Bucket Terminology", May 2000. http://qbone.internet2.edu/bb/Traffic.pdf.

[23] S. Floyd, V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks", IEEE/ACM Transactions on Networking, Vol.3 No. 4, August 1995.

[24] S. Floyd, M. F. Speer, "Experimental Results for Class-Based Queueing", Nov. 1998.

[25] Sally Floyd, "Notes on CBQ and Guaranteed Services" http://www.aciri.org/floyd/papers/guaranteed.ps

[26] D. Clark, S. Shenker, L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism", Proc. SIGCOMM'92, pp.14-26, Aug. 1992.

[27] A. Demers, S. Keshav, S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm", Internet Working: Research and Experience, Vol. 1, pp. 3-26.

[28] Stef Coene: http://www.docum.org/

[29] Fulvio Risso and Panos Gevros, "Operational and Performance Issues of a CBQ router", ACM CCR 1999.

[30] Fulvio Risso, "Decoupling Bandwidth and Delay Properties in Class Based Queueing", ISCC, 2001.

[31] CBQ references. http://www.icir.org/floyd/cbq.html

[32] HTB reference. http://luxik.cdi.cz/~devik/qos/htb/

[33] M. Shreedhar and G. Varghese, "Efficient Fair Queuing using Deficit Round Robin" IEEE/ACM Transactions on Networking, vol.4, no.3, June 1996, pp.375-85, Publisher: IEEE; ACM, USA.

[34] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "A Framework for Internet Traffic Engineering", Internet draft, Jan. 2000

[35] E. C. Rosen, A. Viswanathan, R. Callon, "Multi-Protocol Label Switching Architecture", Internet Draft, 1999.

[36] L. Andersson, P. Doolan, N. Feldman, A. Fredette and B. Thomas, "LDP Specification" RFC3036, Jan. 2001

[37] D. Awduche, L. Berger, D. Gan, T. Li and V. Srinivasan, G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels" RFC 3209. Dec.2001

[38] Chuck Semeria, "RSVP Signaling Extensions for MPLS Traffic Engineering" Juniper Networks, white paper

[39] Paul Brittain, Adrian Farrel, "MPLS Traffic Engineering: A Choice of Signalling Protocols", Data Connection, white paper, Jan. 2000

[40] RFC 2702 "Requirements for Traffic Engineering Over MPLS"

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

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

[43] Anwar Elwalid, Cheng Jin, Steven Low, Indra Widjaja, "MATE: MPLS Adaptive Traffic Engineering" IEEE INFOCOM 200

[44] Torsten Braun, Manuel Guenter, and Ibrahim Khalil. "Management of Quality of Service Enabled VPNs". IEEE Communication Magazine, pp. 90-98, May 2001.

[45] Paul Brittain, Adrian Farrel "MPLS Virtual Private Networks" Data Connection, white paper, Nov. 2000.

[46] N.G.Duffield, Pawan Goyal, Albert Greenberg, Partho Mishra, "A Flexible Model for Resource Management in Virtual Private Networks", ACM SIGCOMM' 99, Oct. 1999, Cambridge, MA, USA.

[47] Chen-Nee Chuah, Lakshminarayanan Subramanian, Randy H. Katz and Anthony D. Joseph. "QoS Provisioning Using A Clearing House Architecture" International Workshop on Quality of Service(IWQoS), Pittsburgh, PA, pp. 115-124, June 2000

[48] Ibrahim khalil, Torsten Braun "Implementation of a Bandwidth Broker for Dynamic End-to-End Capacity Reservation over Multiple Diffserv Domains", 4th IFIP/IEEEInternational Conference on Management of Multimedia Networks and Services (MMNS), Chicago, USA, Oct 29 - Nov 1, 2001

[49] Ibrahim khalil, Torsten Braun "Edge Provisioning and Fairness in VPN-Diffserv Networks". The 9th International Conference on Computer Communication and Network (ICCCN 2000), October 16-18, 2000, Las Vegas, USA.

[50] J. Moy, "OSPF Version 2", RFC 2178, Apr. 1998

[51] D. Oran, "OSI IS-IS Intra-domain Routing Protocol", RFC 1142, Feb.1990.

[52] Y. Jia, M. Guerrero, O. Kabranov, D. Makrakis and L. Barbosa, 'Dynamic Resource Allocation in QoS-Enabled / MPLS Supported Virtual Private Networks and its Linux based Implementation", CCECE'02, Winnipeg, Manitoba, Canada, May 2002.

[53] Y. Jia, M. Guerrero, O. Kabranov, D. Makrakis and L. Barbosa, 'Design and Testbed Implementation of Adaptive MPLS-DiffServ Enabled Virtual Private Networks", CCECE'03, Montreal, Canada, May 2003.

[54] T. Yang, Y. Jia, O. Kabranov, and D. Makrakis, "A New Service Model for Differentiated Services Architecture", 21st Biennial Symposium on Communications, Kingston, Ontario, Canada, June 2002

[55] IBCN's RSVP-TE for Diffserv over MPLS, http://dsmpls.atlantis.rug.ac.be/

[56] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol" RFC 2401, Nov.1998.

[57] R. Rivest, "The MD5 Message-Digest Algorithm", RFC1321, April 1992

[58] F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen etc. "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services" RFC3270, May 2002

[59] J. R. Gallardo, D. Makrakis, L. Orozco-Barbosa. "Prediction of Alpha-Stable Long-Range Dependent Stochastic Process", Advances in Performance Analysis, 3(1), pp. 31-42, 2000.

[60] J. R. Gallardo, D. Makrakis, L. Orozco-Barbosa, "Use of Alpha-Stable Self-Similar Stochastic Process for Modeling Traffic in Broadband Networks". Performance Evaluation, 40(1-3), pp. 71-98, 2000.

[61] R. Law, S. Raghavan, "Diffserv and MPLS – Concepts and Simulation"

[62] AMEsberger, "Linux Traffic Control - Implementation Overview", Technical Report SSC/1998/037, EPFL, November 1998 ftp://lrcftp.epfl.ch/pub/people/aMEsber/pub/tcio-current.ps.gz.

[63] TC API: http://oss.software.ibm.com/developerworks/projects/tcapi

[64] W. Richard Stevens, "UNIX Network Programming", volume 1, second edition, Prentice Hall, 1998

[65] Torsten Braun, Matthias Scheidegger, Hans Joachim Einsiedler. "A Linux Implementation of a Differentiated Services Router"Networks and Services for Information Society (INTERWORKING'2000), October 3-6, 2000, Bergen, Norway

[66] D. Verma, "Supporting Service Level Agreements on IP networks" Macmillan Technical Publishing, 1999

[67] E. Clark, " SLAs: In Search of the Performance Payoff ", Network Magazine, May 1999

[68] Chen-Nee Chuah "A Scalable Framework for IP-Network Resource Provisioning Through Aggregation and Hierarchical Control" Ph.D thesis, chapter 5. http://www.cs.berkeley.edu/~chuah/thesis/thesis_summary.html.

[69] A. Leon-Garcia "Probability and Random processes for Electrical Engineering", Second Edition. Addison-Wesley Publishing Company.

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

[71] Tarek Saad, "Support of Delay Densitive Applications over MPLS and Diffserentiated Services enabled Networks". Master thesis.

[72] J. Postel, "TRANSMISSION CONTROL PROTOCOL", RFC793, Sept.1981.

[73] P. Deleon and C. J. Sreenan, "An Adaptive Estimator for Media Playout Buffering" in Proceedings of IEEE ICASSP, Phoenix, AZ, March 1999, pp. 3097-3100

[74] K. Park, W. Jeon, T. Basar and C. Choi, "Robust Delay Estimation for Internet Multimedia Applications" IDMS/PROMS 2002, LNCS 2515, pp.255-262, 2002

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

[76] "Tcpdump and Libcap ", http://www.tcpdump.org/

[77] "Querying libiptc HOWTO", http://opalsoft.net/qos/libiptc/qlibiptc.html

[78] Werner AMEsberger, "Linux Network Traffic Control—Implementation Overview" Feb. 2001

[79] Werner AMEsberger, et al. "Differentiated Services on Linux," Internet Draft < draft-aMEsberger-wajhak-diffserv-linux-01.txt>, June 1999

[80] "Differentiated Services on Linux" http://diffserv.sourceforge.net/

[81] Bert hubert, Gregory Maxwell, et al. "Linux Advanced Routing & Traffic Control HOWTO". Published v0.9.0 Date: 2002/03/06.

[82] "InterWatch 95000 Protocol Analyzer user guide" by GN Nettest

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

[84] ISO/IEC "The MPEG Home Page" http://mpeg.telecomitalialab.com

[85] Rob Koenen, " Overview of the MPEG-4 standard " (V.21) http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm

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

[87] "QuickTime Streaming Server," http://www.apple.com/quicktime/products/qtss/

[88] "RealNetworks," http://www.realnetworks.com

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

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

[91] Beran, J. Sherman, R., Taqqu, M. S., Willinger, W., "Long-Range Dependence in Variable Bit Rate Video Traffic ", IEEE Transactions on Communications, Vol. 43, No. 2-4, pt. 3, pp.1566-1579, 1995.

[92] Vern Paxson, and Sally Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling", IEEE/ACM Transactions on Networking, Vol. 3. No.3, June 1995

[93] Jim W. Roberts, "Traffic Theory and the Internet", IEEE Communications Magazine, Jan. 2001, pp. 94-99.

[94] D. P. Heyman and T. V. Lakshman, "What Are the Implications of Long-Range Dependence for VBR-Video Traffic Engineering?" IEEE/ACM Transactions on Networking, Vol. 4. No.3, June 1996

[95] J. R. Gallardo, "Fractional Stable Noise Processes and Their Application To Traffic Modeling and Fast Simulation of Broadband Telecommunications Networks", PHD thesis.

[96] V. Frost and B. Melamed, "Traffic Modeling for Telecommunications Networks", IEEE Communications Magazine, Vol. 33, pp.70-80, Mar. 1994

[97] W. Leland, M. Taqqu, W. Willinger and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (extended version)", IEEE/ACM Transactions on Networking, Vol. 2, pp. 1-15, Feb.1994

[98] Simon Pietro Romano, Marcello Esposito, Giorgio Ventre, Giovanni Cortese "Service Level Agreements for Premium IP Networks" draft-cadenus-sla-00.txt

[99] Cisco White Paper " Cisco MPLS AutoBandwidth Allocator for MPLS Traffic Engineering: A Unique New Feature of Cisco IOS Software" http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/mpatb_wp.htm

[100] A. M. Zoubir and B. Boashash, " The bootstrap and its applications in signal processing", IEEE Signal Processing Magazine, Jan. 1998

[101] Iptables-1.2.6 by Paul Russell. http://www.netfilter.org/

[102] Linux Netfilter Hacking HOWTO by Paul Russell. http://www.netfilter.org/documentation/HOWTO/

[103] Querying Libiptc HOWTO. http://opalsoft.net/qos/libiptc/qlibiptc.html

[104] A. Silberschatz and P. Galvin, " Operating System Concepts" Fifth edition. pp. 155-177

[105] Pim Van Heuven et.al "RSVP-TE daemon for DiffServ over MPLS under Linux Features, Components and Architecture" Linux-Kongress 2002, 9th International Linux System Technology Conf. Sept. 4-6, Germany

[106] Matthew G. Marsh, "Policy Routing With Linux - Online Edition" http://www.policyrouting.org/PolicyRoutingBook/ONLINE/TOC.html

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

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

[109] S. M. Lei, T. C. Chen, and M. T. Sun, "Video Bridging Based on H.261 Standard," IEEE Trans. on Circuit and System. For Video Tech., vol. 4, no. 4, Aug. 1994, pp. 425-437.

[110] ITU-T Recommendation H.263 (1995): "Video Coding for Low Bit Rate Communication".

[111] Project "Charging and Accounting Technologies for the Internet (CATI)", http://www.tik.ee.ethz.ch/~cati/home.html

[112] MPLS for Linux project: http://sourceforge.net/projects/mpls-linux

[113] A. Karasaridis, D. Hatzinakos, "Network Heavy Traffic Modeling Using α-Stable Self-Similar Processed" IEEE Trans. On Communications, Vol.49, No.7, July 2001.

[114] Miguel López Guerrero, PHD Dissertation, "On Resource Allocation Technique Using Traffic Models Based on α-stable Stochastic Process"

[115] Miguel López Guerrero, Luis Orozco Barbosa, Dimitris Makrakis, "On the Use of Probabilistic Envelope Processes In Resource Allocation" CCECE 2003, Montréal, Canada, May 2003

[116] Fotios C. Harmantzis, Dimitrios Hatzinakos, Irene Katzela, "Shaping and Policing of Fractal a-stable Broadband Traffic". CCECE 2001, IEEE Canada, May 13-16, 2001, Toronto

APPENDIX A1
VALIDATION OF MGEN TRAFFIC GENERATOR

The objective of this experiment is to validate the MGEN [83] traffic generator. MGEN can generate two traffic patterns: Poisson and Periodic. For each case, we use MGEN to generate different traffic rates like 20, 50, 100, 150, 250 packets/second. The packet size is fixed at 1000 bytes. We capture the traffic at the output link of the Traffic Generator by using an external network analyzer. For Poisson traffic, we perform the following tests:

- Calculate the average traffic rate of measured traffic λr to confirm that the produced traffic has volume equal to the one the system intended to produce (λs).

- Record and analyze the inter-arrival times of packets τ. Confirm it follows the exponential distribution with the proper statistics.

- Calculate the average packet inter-arrival time τ. There is a relation between average traffic rate λ and average packet inter-arrival time: $\tau = (1/\lambda)$

- One of the key characteristics of Poisson traffic is that it does not show correlation. The generation of each packet is completely independent from the generation of any other packet, the previous and next ones included. We assume *N* packets, we have a sequence of *n* inter-arrival times, *IT(i)* ($1 \le i \le n$, n = *N-1*). Subtract from all values the average inter-arrival time τ. Assume $X(i) = IT(i) - \tau$, then, calculate the normalized sample autocorrelation function of the sequence *X(i)* [113].

$$\hat{p}(k) = \frac{\sum_{i=1}^{n-k} X(i)X(i+k)}{\sum_{i=1}^{n} X^2(i)} \qquad \text{(A-1)}$$

The result should show $\hat{p}(0) = 1$, other $\hat{p}(k)$ should be close to 0.

Table A-1 shows the validation results for Poisson traffic. Figure A-1 to Figure A-5 provide the analytical and experimental inter-arrival time for various traffic volumes (from values of 20 packets/second to 250 packets/second). The results indicate as far as first and second order statistical coverage, autocorrelation functions, the behavior is close to the expectation. Some more noticeable deviation appears for speeds higher than 100 packets/second. However, from Figure A-1 to A-5, we see that there is quite noticeable deviation of the measured packet inter-arrival time from the expected results. We notice that the deviations appear more pronounced at the lower values of inter-arrival times, and the deviation becomes more serious as we move to higher volumes of traffic. For example, for speeds higher than 100 packets/second, there is strong concentration at very small values of inter-arrival time and then, a higher concentration from what expected at higher values. Nevertheless, in all cases we see noticeable deviations at the smaller values region. We believe that the source of this behavior might be the following two reasons. First, we are not running Linux in real-time mode. Because of this, a number of processes other than the traffic generator can capture the resources of PC during a period the generator might require them. It is quite evident that such a problem might be more noticeable in cases where packets are generated close to each other. As the processing of a packet generation completes, other processes waiting for service capture resources. Thus, if a new packet has to be generated, it has to wait until the service of those processes complete before it becomes served.

The second potential source of deviation is the inability of the PC to "push" the packets out of its interface, to the network in time. Again, many processes are running at the PC. There is chance that resources are taken away from the I/O process of the PC. As packets are been "stuck", unable to follow the previous packet in time, they experience long inter-arrival time. Now as such packets become "queued"; waiting for service, when the resource releases, they are processed in a burst, generating the very small inter-arrival time concentrations. We can see in the figures. It is quite evident that this behavior becomes exemplified as the traffic volume becomes higher, producing higher pressure on the resources of the PC. Thus, we see a trend of deviation from Poisson to a burstier traffic behavior. However, based on the observations, we conclude that up to 100 packets/second, the generator is capable of producing reasonable accurate statistics.

Table A-1. Validation of Poisson traffic

λs (packets/second)	λr (packets/second)	$1/\lambda r$	τ (second)
20	21	0.0476	0.051970147
50	52	0.0193	0.019740891
100	102	0.009793	0.009885934
150	153	0.006511	0.0066568
250	260	0.003841	0.003965

λs	Autocorrelation function		
(packets/second)	$\hat{p}(0)$	$\hat{p}(1)$	$\hat{p}(2)$
20	1	0.004754385	-0.031294116
50	1	0.003444483	-0.020564797
100	1	-0.058511369	-0.020740703
150	1	-0.121296657	-0.026617857
250	1	-0.200311422	-0.049133621

We also evaluate the Periodic traffic (CBR) pattern generated by MGEN. We repeat the same experiments as for Poisson. For CBR, we only analyze the packet inter-arrival time distribution. Figure A-6 to A-10 show the inter-arrival time distribution with different traffic rates. It is evident that the deviations from the expected value of inter-arrival time become more pronounced as the traffic volume increases. The deviations appear at both sides (smaller and larger) of the expected value, indicating that source of the problem is the reasons discussed earlier for Poisson. It is also clear that due to such effects, the traffic becomes burstier than what it was originally intended to be.

Been concern with the effect, the functionality of the PCs might have on the statistics of the produced traffic, we conduct a measurement at the output of the Ingress router for the 100 packets/second stream to access how close the traffic remains to Poisson, after passing through the generator and Ingress router. The result, shown in Figure A-11, indicates there is reasonable agreement between experimental results and expected behavior (analytical value).

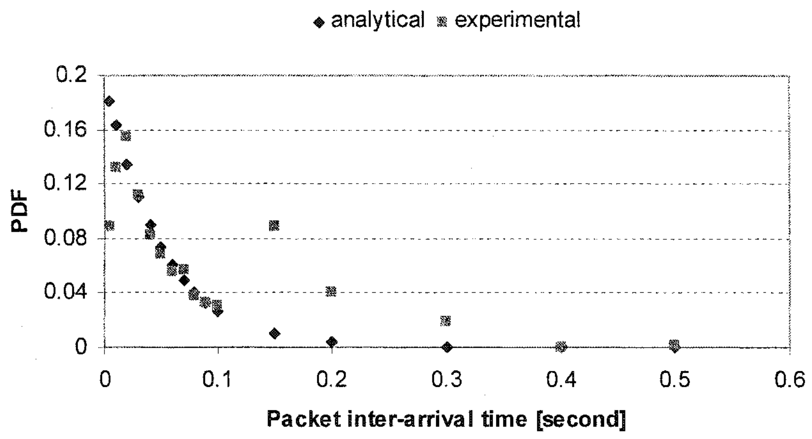

Figure A-1. Exponential distribution of inter-arrival time of Poisson traffic with rate 20 packets/second

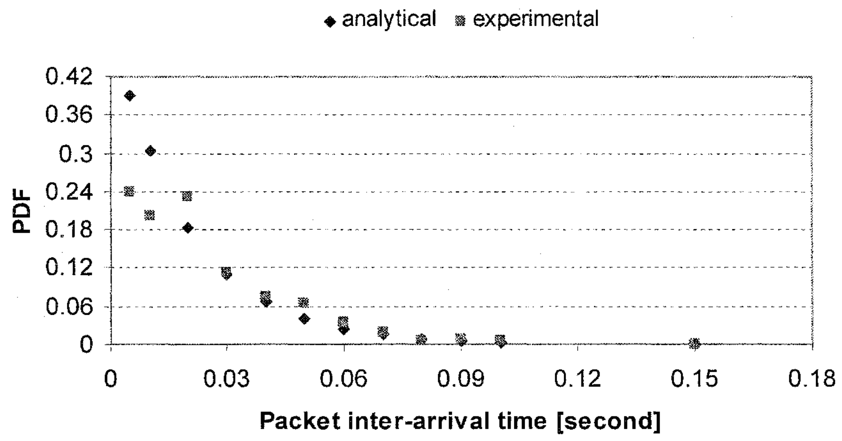

Figure A-2. Exponential distribution of inter-arrival time of Poisson traffic with rate 50 packets/second

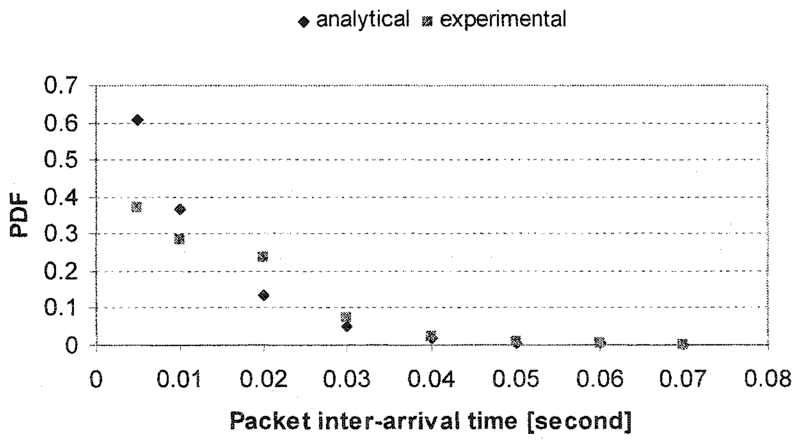

Figure A-3. Exponential distribution of inter-arrival time of Poisson traffic with rate 100 packets/second

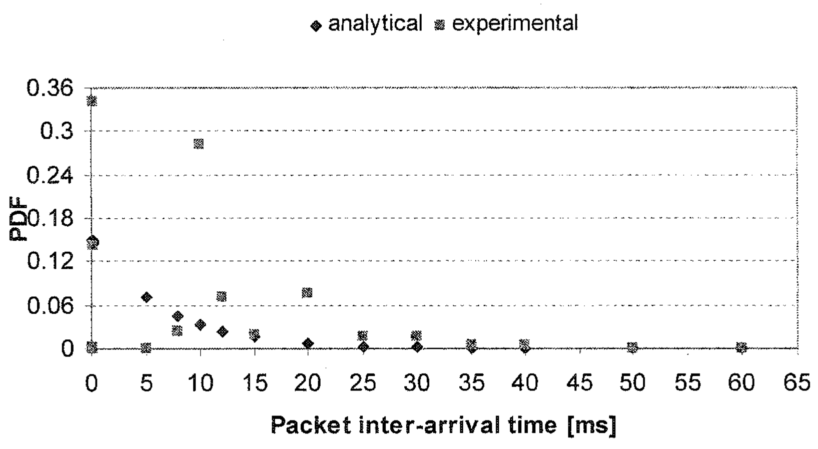

Figure A-4. Exponential distribution of inter-arrival time of Poisson traffic with rate 150 packets/second

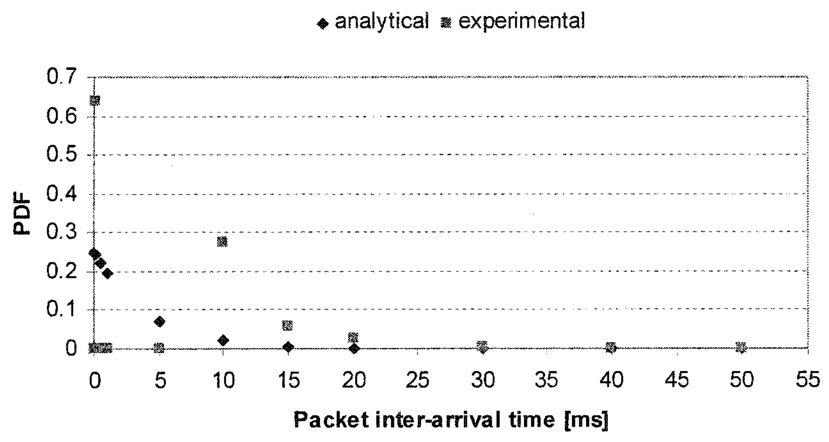

Figure A-5. Exponential distribution of inter-arrival time of Poisson traffic with rate 250 packets/second

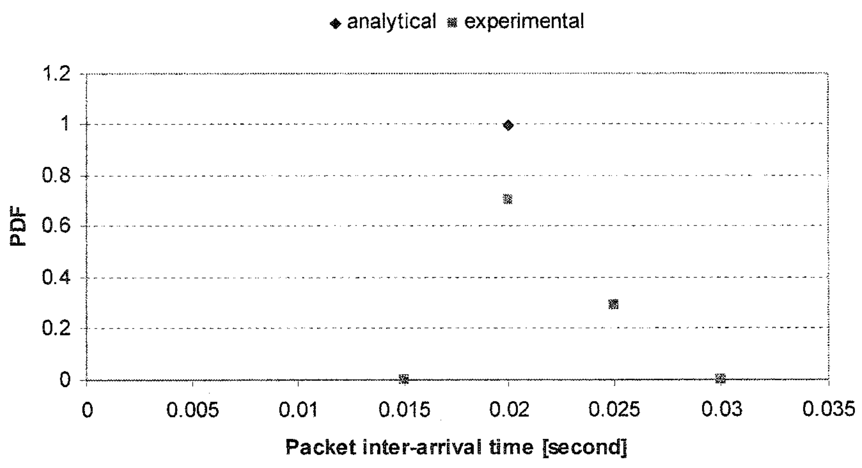

Figure A-6. Inter-arrival time distribution of CBR traffic with rate 50 packets/second

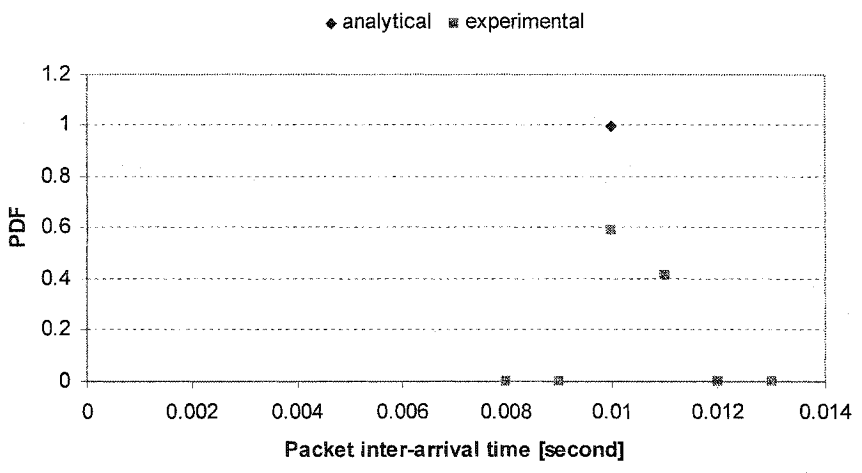

Figure A-7. Inter-arrival time distribution of CBR traffic with rate 100 packets/second

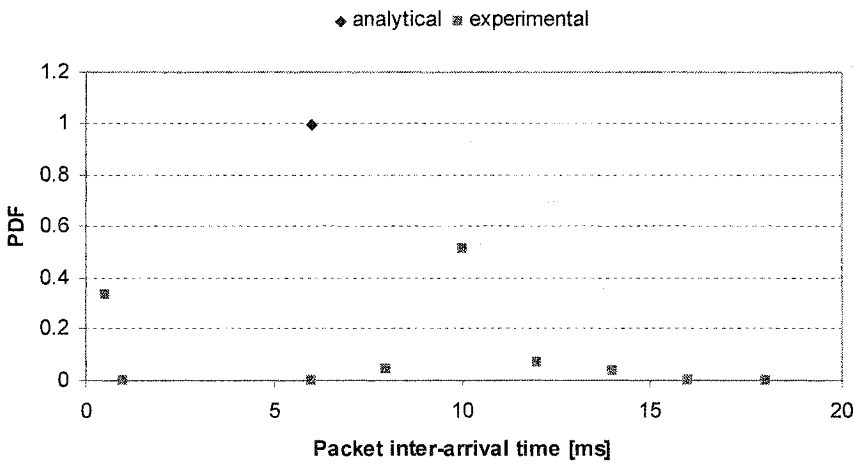

Figure A-8. Inter-arrival time distribution of CBR traffic with rate 150 packets/second

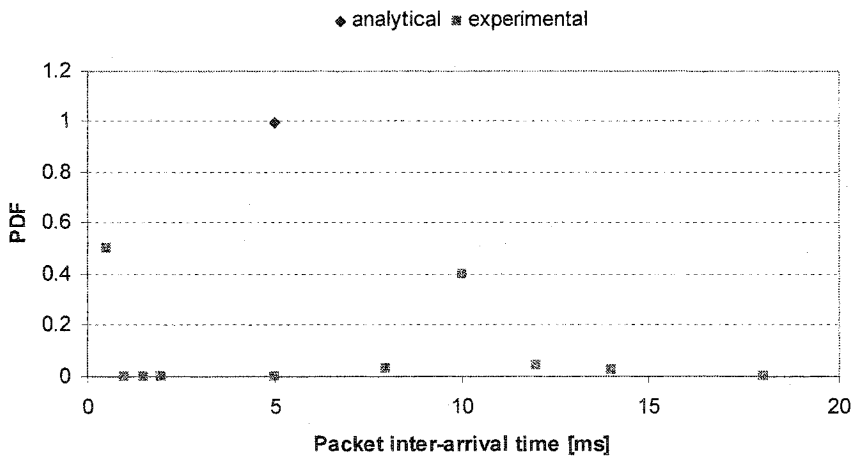

Figure A-9. Inter-arrival time distribution of CBR traffic with rate 200 packets/second

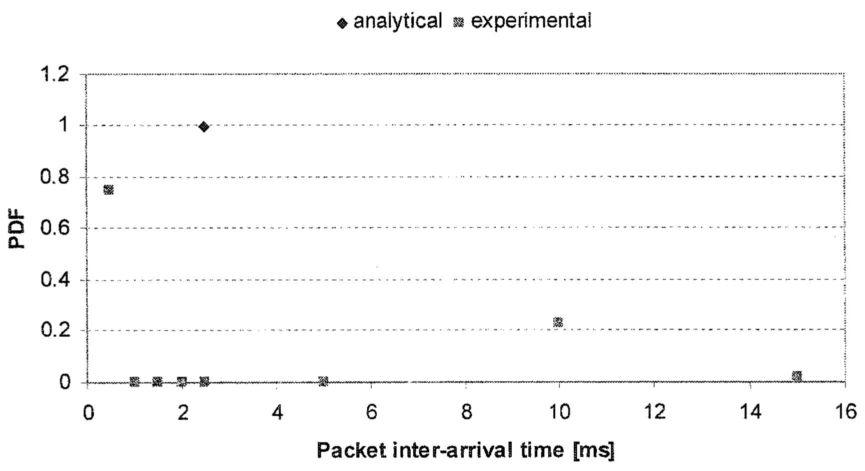

Figure A-10. Inter-arrival time distribution of CBR traffic with rate 400 packets/second

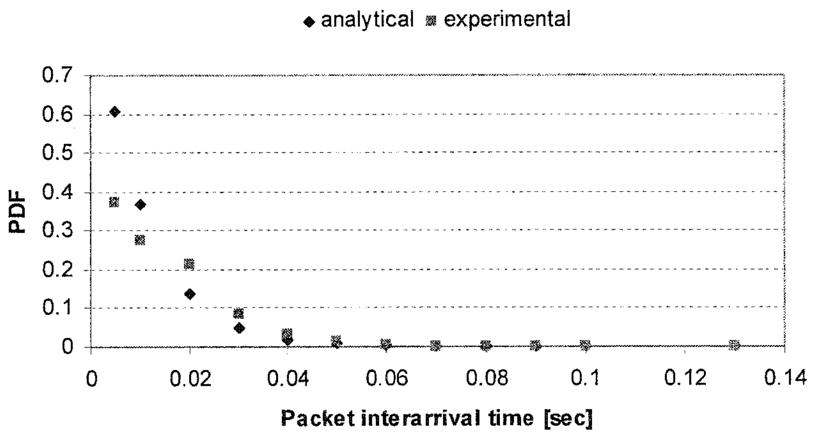

Figure A-11. Exponential distribution of packet inter-arrival time of Poisson traffic with rate 100 packets/second after passing through Ingress router in IP mode

APPENDIX A2
VALIDATION OF α–STABLE LONG-RANGE DEPENDENT TRAFFIC GENERATOR

There are five types of traffic able to be generated by our α–stable long-range dependent traffic generator [60]. We focus on the MPEG-1 video type of traffic. The parameters we considered are: α, H. α is called the index of stability that presents the variability of traffic. H is called the Hurst parameter (or index of self-similarity). Table A-2 shows the validation results for α–stable MPEG-1 traffic. Figure A-12 and Figure A-13 is one example of comparison between the original trace (before sending to the network) and the recorded trace (recorded at the output link of the generator) with mean rate 50 packets/second and $\alpha = 1.95$. Figure A-14 and Figure A-15 shows another example with $\alpha = 1.60$ that is burstier traffic. It proves our generator works well. Please note that when we deploy our dynamic bandwidth allocation scheme, since the number of samples within the small observation window is far smaller than the number of samples of the entire sequence, it is a truncated α–stable process, the value of α will be a little different from that of the entire sequence. Because of the processing limitation of the Host PC, the maximum traffic rate we can generate without deviation from the expected statistical behavior is 50 packets/second.

Table A-2. Validation of α–stable MPEG-1 traffic

Traffic rate	Model parameters		Validation parameters	
(packets/second)	α	H	α	H
20	1.95	0.903	1.94	0.91
50	1.95	0.903	1.93	0.904
20	1.60	0.903	1.61	0.902
50	1.60	0.903	1.62	0.903

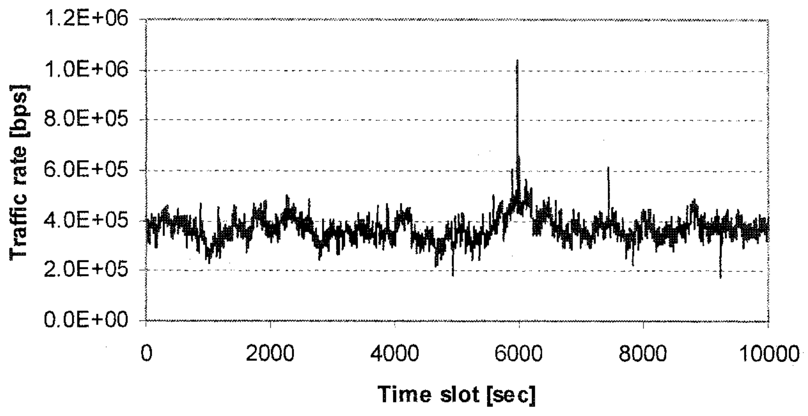

Figure A-12. Original trace of α–stable with MPEG-1 characteristics traffic based on the model (α=1.95, H = 0.903)

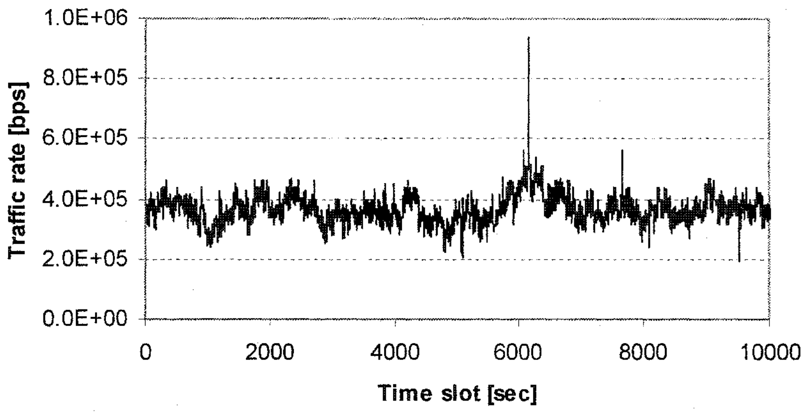

Figure A-13. Recorded trace of α–stable with MPEG-1 characteristics traffic (α =1.94, H=0.904)

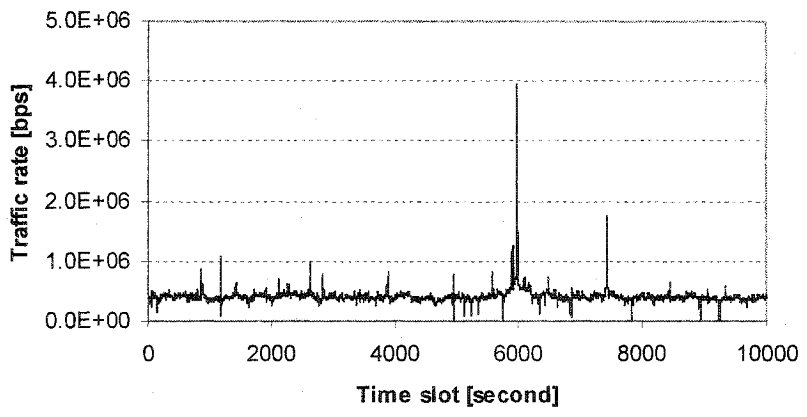

Figure A-14. Original trace of α–stable with high variability VBR traffic based on the model (α=1.60, H = 0.903)

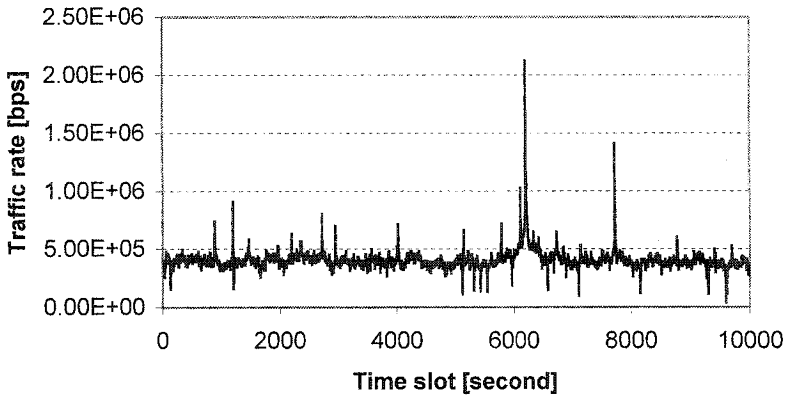

Figure A-15. Recorded trace of α–stable with high variability VBR traffic (α =1.62, H=0.903)

APPENDIX B
VALIDATION OF THE MPLS SWITCHING PROCESS

The objectives of this testing are:

- Assess the ability of the Linux-built router to provide the benefits of MPLS fast switching.

- Find out up to what traffic levels the MPLS router can function properly. It means identify the level of traffic where a router starts becoming the bottleneck. There are two factors that could result in bottleneck. One is the network resource. The other is the CPU processing power. In our network topology, the Core router is the bottleneck (output link speed 10 Mbps). However, sometimes the router cannot forward packets in time because it does not have enough processing power to execute the calculation fast enough.

We run the system in both, the Best-effort and Diffserv configurations.

B1: Best Effort Configuration over IP and MPLS network

The objective of the experiments is to explore the MPLS forwarding capability compared with normal IP forwarding paradigms when using with best-effort configuration. The default buffer size is set to 100 packets in each router. We run experiments in IP mode and MPLS mode separately. For each case, we calculate the average packet forwarding delay at each router with background Poisson traffic. The rate is chosen 1, 3, 5, 7, 8, 10 Mbps. Poisson traffic is generated by MGEN, the packet size is fixed at 1000 bytes.

The network topology is shown in Figure 4-1. We use a common reference analyzer to measure per node delay or loss to avoid clock synchronization problem as indicated earlier. Tethereal is used to measure the delay for MPLS packet. Tcpdump is for IP packet. We repeat the same experiment 10 times in order to get the accurate statistics. The average number of packet sending through the network is 50,000.

Figure B-1 and Figure B-2 shows the average forwarding delay statistics at each router in IP and MPLS mode. Observe that in IP mode, the Ingress and Egress routers show similar forwarding behavior, however, the delay for the Core router increases sharply when the traffic rate is over 70% network utilization (please be reminded that Core router is the bottleneck, 10 Mbps link capacity). In MPLS mode, Ingress behaves differently as compared to the IP mode. The delay for Ingress increases sharply when the traffic rate is over 6 Mbps. The possible reason is that the Ingress router experiences more overhead in MPLS mode.

Figure B-3 to Figure B-5 demonstrate each router's behavior when deployed in IP/MPLS mode. Observe that Ingress shows different behavior under IP/MPLS mode. The Core and Egress shows similar behavior under these two modes. In theory, MPLS forwarding should faster than IP forwarding. However, results shown here do not give us clear advantage of MPLS forwarding. This can be contributed to the fact that MPLS has been implemented in the form of software rather than hardware, thus, the fast label processing capabilities diminish.

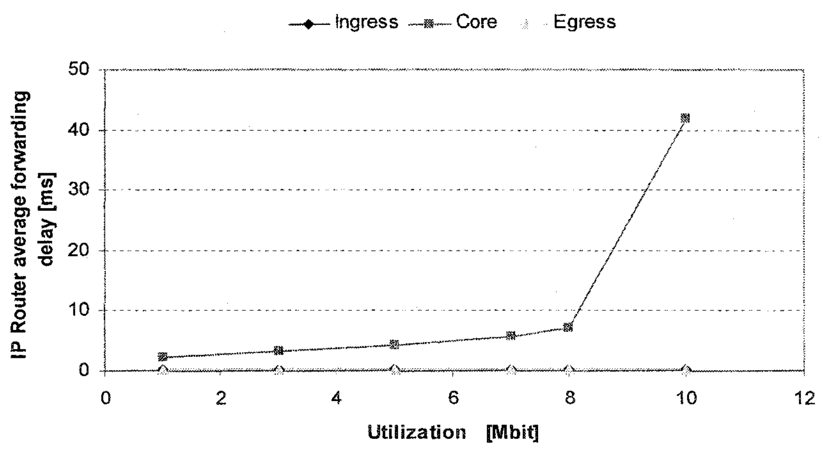

Figure B-1. Average packet forwarding delay vs. utilization at each router in IP mode

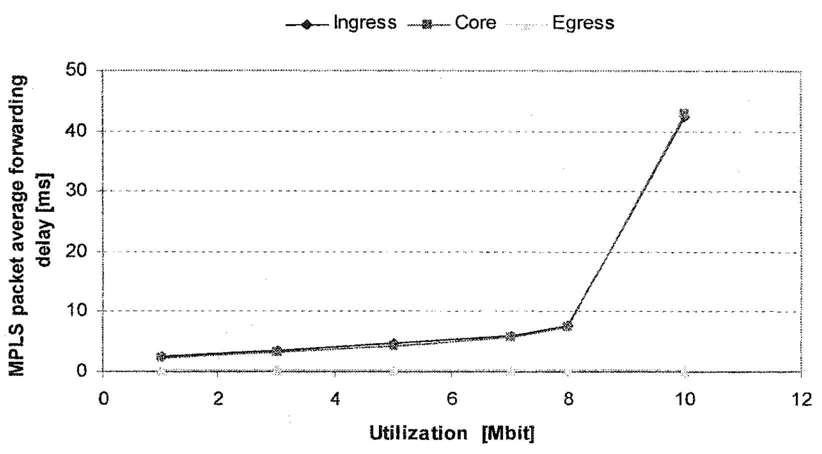

Figure B-2. Average packet forwarding delay vs. utilization at each router in MPLS mode

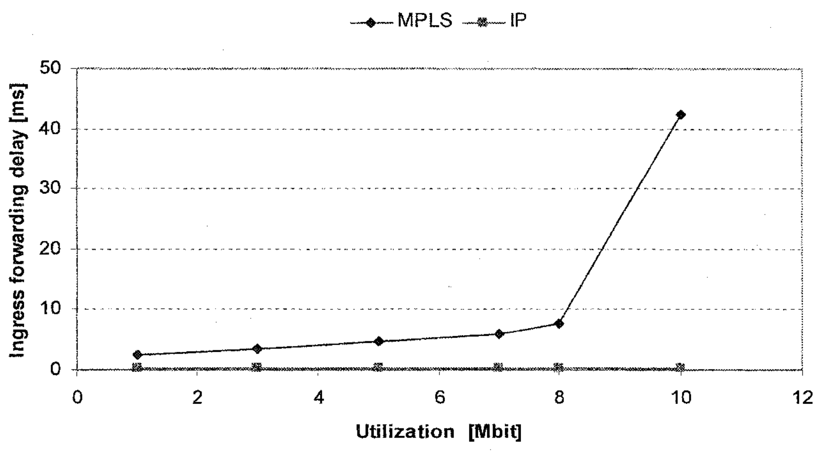

Figure B-3. Ingress average packet forwarding delay vs. utilization for a) MPLS mode, b) IP mode

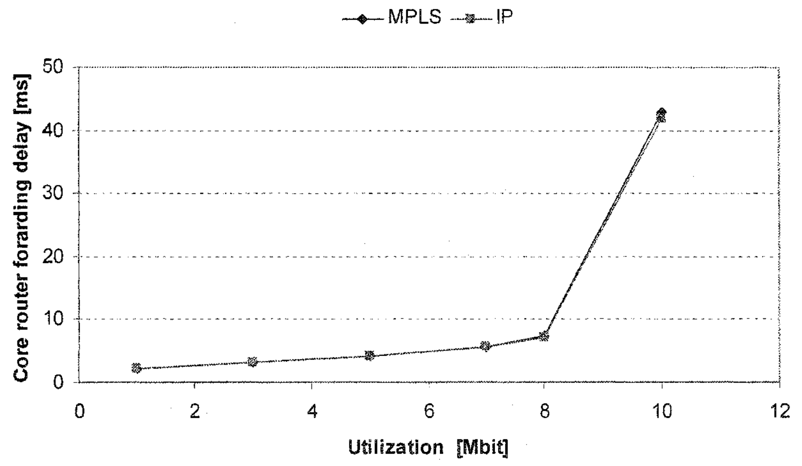

Figure B-4. Core average packet forwarding delay vs. utilization for a) MPLS mode, b) IP mode

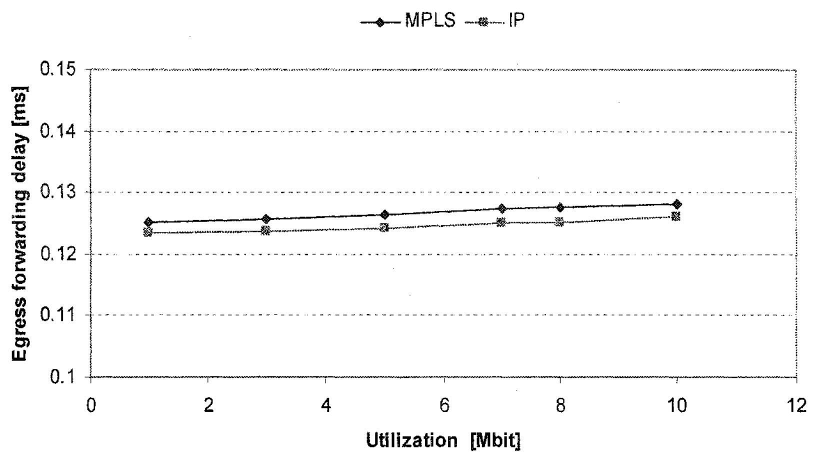

Figure B-5. Egress average packet forwarding delay vs. utilization in a) MPLS mode, b) IP mode

B2: Diffserv Configurations over MPLS

In this experiment, we want compare the performance between MPLS Diffserv and no Diffserv. We specify two types of traffic: High priority MPEG-2 stream video with mean rate 2.6 Mbps and background traffic. The background traffic is the CBR traffic generated by MGEN [83]. The video server and video client are set up based on VideoLan [89]. We deploy CBQ with EF rate 4M, buffer size 20 packets. We send MPEG-2 video stream through the network, increasing the background traffic utilization (the total is 10M). The packet size of CBR traffic is 512 bytes.

Again we use a common reference analyzer to measure the average one-way delay or loss in order to avoid the clock synchronization problem. Figure B-6 and Figure B-7 show average one-way delay and average packet loss rate of MPEG-2 video stream with MPLS Diffserv and no Diffserv. Observe that the average one-way delay is almost the same (from 2.7ms to 4.4ms) for MPLS Diffserv case, but the delay increases sharply when background traffic is above 70% of bottleneck utilization (from 2.7ms to 62ms with 100% load) with no Diffserv. The average loss rate shows similar behavior. It is always 0% for the MPLS Diffserv case, yet, it changes from 0% to 27% for the no Diffserv case. Figure B-8 shows the IPDV jitter experienced by the EF packets. Observe that the jitter is always the same (1.25ms), no mater how much the background traffic increases. It proves that MPLS Diffserv can guarantee the network performance, especially under network congestion.

We also conduct experiments with different values of packet size for the background traffic for the MPLS Diffserv case. The used packet sizes are 128, 256, 512, 1024, 1280 and 1408 bytes. Figure B-9 and Figure B-10 show the average one-way delay and IPDV jitter experienced by the EF traffic with different BE packet size when 100% CBR background traffic utilization. Observe that when the packet size increases, the transmission time of packet increases, so the total end-to-end delay increases as well. The IPDV jitter keeps almost same value (1.25ms)

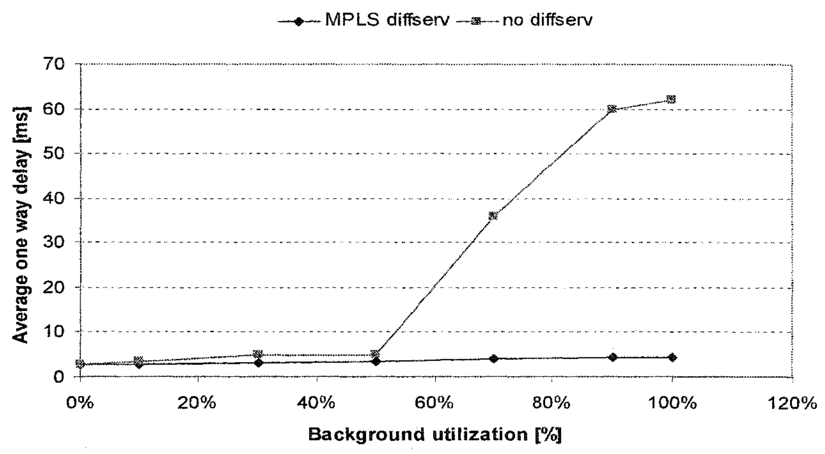

Figure B-6. Average one-way delay of MPEG-2 video stream [ms] vs. CBR background traffic utilization for (a) MPLS Diffserv with CBQ EF rate 4M, buffer size of 20 packets, (b) No Diffserv

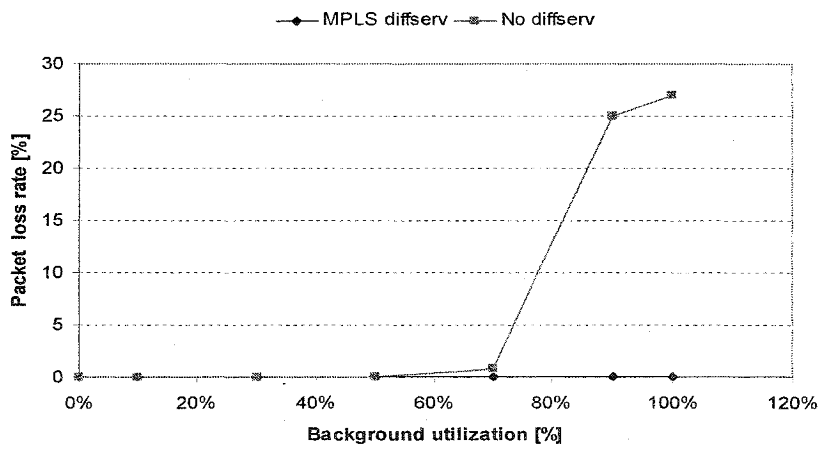

Figure B-7. Average loss rate of MPEG-2 video stream vs. CBR background traffic utilization for (a) MPLS Diffserv with CBQ EF rate 4M, buffer size of 20 packets, (b) No Diffserv

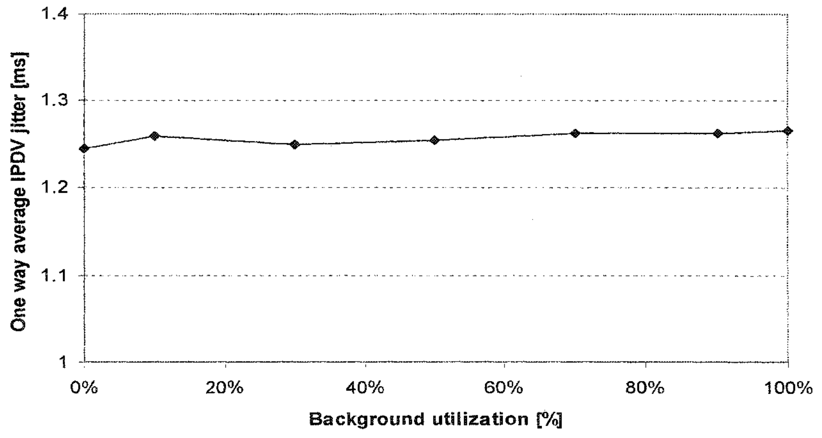

Figure B-8. IPDV jitter [ms] of MPEG-2 video stream vs. CBR Background utilization when MPLS Diffserv path with CBQ EF 4M, buffer size of 20 packets

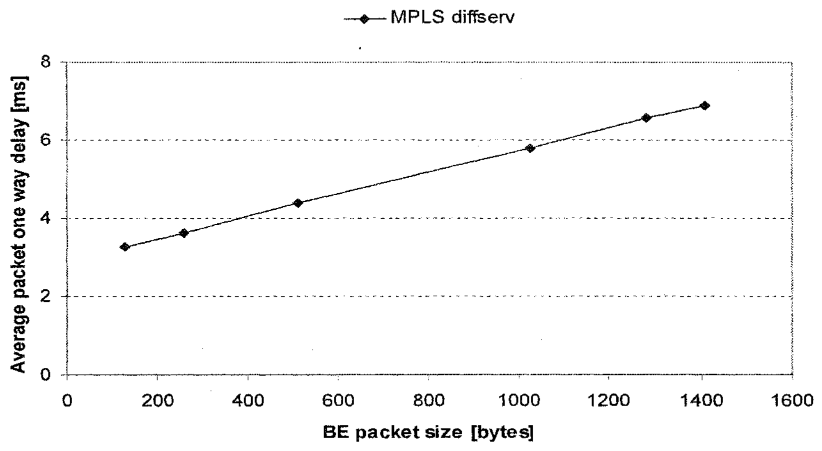

Figure B-9. Average one-way delay [ms] of MPEG-2 video stream vs. CBR background traffic packet size when MPLS Diffserv path with CBQ EF 4M, buffer size of 20 packets

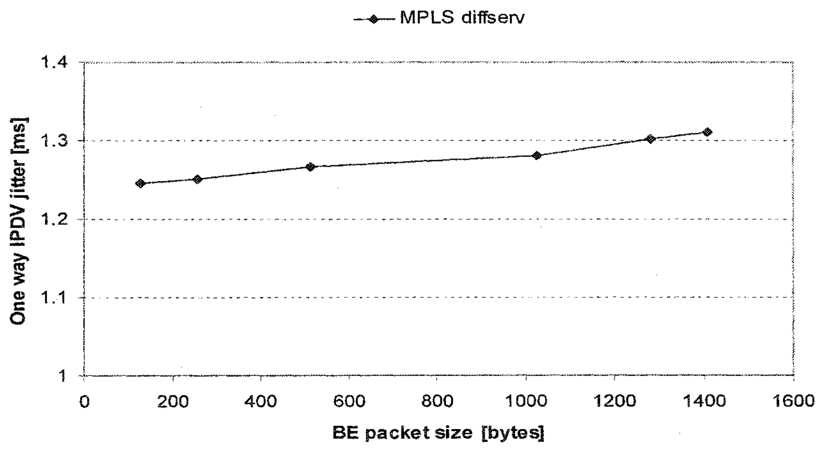

Figure B-10. IPDV jitter [ms] of MPEG-2 video stream vs. CBR background traffic packet size when MPLS Diffserv path with CBQ EF 4M, buffer size of 20 packets

B3: Performance Issues

MPLS transforms the problem of finding the next hop in the forwarding table using an exact match based on the destination IP address or a prefix to the problem of finding an exact match of the label. Typical, IP route looking up is expensive. This is totally unacceptable for high speed routing where several terabits of data has to move in a second. Here comes the MPLS to our rescue. It has nothing to do with the routing protocols, but how forwarding information is looked from the forwarding tables.

With a label in a packet, we can now have exact match look up - in an index array or a hash table. This means that we can do search in O(1) time. Moreover, the search time is deterministic. Therefore, we can have route look up much faster and in deterministic times. This is what MPLS is all about.

But it is not going to see any performance difference with a small MPLS network. By creating a hundred thousand entries in forwarding table and sending some high volume data - the effect will become vary clear. However, the relative performance analysis depends on the details of a specific implementation. For example, our Linux test-bed is based on the James's MPLS forwarding code [112]. The goal of this project is not to provide fast forwarding compare with some commercial product.

New technology has made it possible for IP routing software to be done as fast as MPLS switching, e.g. hardware implementation, pipelined ASICs, hashing, advanced searching techniques, routing table compression. Some Core vendors have built necessary pipelined ASICs that can do wire speed forwarding at OC48 rates, or perhaps even at better speeds with a forwarding table of over 50,000 plus prefixes. Some of the latest search techniques such as controlled prefix expansions in hardware are done. All of this makes the relevance of this observation on MPLS network performance rapidly diminish even that in theory, MPLS switching is faster than IP forwarding because it's based on table indexing and not a longest-prefix match.

It has been proven that the strongest points of MPLS aren't forwarding performance but the network traffic engineering capabilities it offers. Forwarding is probably not a key influence on "MPLS performance" overall. A worthwhile comparison is really to examine what MPLS provides in terms of traffic engineering capabilities relative to either pure IP routing, or IP routing augmented with some traffic engineering capabilities in the interior gateway protocols (IGP), such as equal-cost multi-path, or metric tuning. MPLS allows non-shortest paths to be established, and utilized - the key to optimal network-wide load balancing. The traffic engineering capabilities of non-shortest path LSPs that probably reduce the operational costs far more over a year than the few cost savings in hardware associated with the label lookup.