u Ottawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Dongli Zhang
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Electrical Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

QoS Control and Guarantees for MPLS VPN Based Services: The Packet Loss Case

TITRE DE LA THÈSE / TITLE OF THESIS

Dan Ionescu
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Abdelmotaleb El-Saddik

Ioanis Nikolaidis

Nicolas Georganas

Chang Cheng Huang (absent)

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

QoS Control and Guarantees for MPLS VPN Based Services: The Packet Loss Case

by

Dongli Zhang

Doctoral Dissertation

Submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa
May 2006

Your file Votre référence
ISBN: 978-0-494-25896-5
Our file Notre référence
ISBN: 978-0-494-25896-5

I hereby declare that I am the sole author of this thesis.

I authorize the University of Ottawa to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Dongli Zhang

I further authorize the University of Ottawa to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Dongli Zhang

ABSTRACT

The market opportunity is tremendous for cost-effective and flexible MPLS VPN services overall. QoS control and guarantees for provided VPN services is a critical issue to be addressed such that service providers can maximize their profits and keep their competitive edge in the market. The focus of this thesis is to solve the quantitative QoS control problem in the provisioning of QoS enabled VPN services across the core IP/MPLS networks, especially in the packet loss case.

A QoS control framework is proposed in a control-theoretic perspective, where the open-loop and closed-loop control techniques are combined together. As one of the key components of the solution devised, a traffic monitor is first designed and implemented in Java. The packet loss estimators calculate the packet loss probability based on traffic measurements, where two estimation formulae are derived by applying the large deviation theory. A reactive estimator is then constructed on-line, which surpasses the original two estimators with the capacity of adapting itself to model imprecisions. Furthermore, a Kalman filter is employed to optimally estimate the packet rate mean and variance. The resulting reactive estimator is applied to the service admission control for admitting an appropriate amount of services and traffic flows into the core network.

The focus of the thesis is on the dynamic packet loss probability control for VPN services. Preliminary controllers are first designed and evaluated for the analytic loss model. System identification technology is then applied to identify the dynamic loss model, where both of the offline and online prediction error methods are explored. Controlling the identified model is examined by both classical and modern controller design technologies. Since the identified linear model contains time-varying and norm-bounded parameters, a Linear Matrix Inequality (LMI) approach for designing robust and optimal controllers is also studied. The performance of the packet loss control system is evaluated with real measured traffic from the live NCIT*net2 network.

ACKNOWLEDGMENT

I wish to express my sincere gratitude to my supervisor, Dr. Dan Ionescu, for his constant guidance, encouragement, and support throughout my research. It would not have been possible to finish this long-term work without his support.

I would also like to express my sincere thanks to the members of my advisory committee and thesis examiners for their encouragement, discussions and suggestions.

I also wish to thank my colleagues and friends, Bogdan Ionescu, Marcel Ionescu, Claudiu Stejarel Veres, Mircea Trifan, Raymond Peterkin and Dr. Cristian Lambiri for their helps and encouragements.

Finally, I would like to thank my family for their support, patients, kindness, inspiration and love.

Contents

List of Tables

List of Figures

Notation

Various symbols, superscripts, subscripts, and abbreviations used frequently in this proposal are summarized below. All symbols are fully defined where first used in the text.

Acronyms and Definitions

AC	Admission Control
AF	Assured Forwarding
AIC	Akaike Information Criterion
AIMD	Additive Increase Multiplicative Decrease
AR	Auto-Regressive
ARE	Algebraic Riccati Equation
ARMA	Auto-Regressive Moving Average
ATM	Asynchronous Transfer Mode
BECN	Backward Explicit Congestion Notification
BGP	Border Gateway Protocol
CAC	Connection Admission Control
CLI	Command Line Interface
CoS	Class of Service
DAC	Direct Admission Control
DiffServ	Differentiated Services
EB	Effective Bandwidth
EF	Expedited Forwarding

EPRCA	Enhanced Proportional Rate Control Algorithm
ERICA	Explicit Rate Indication for Congestion Avoidance Algorithm
ETE	End-to-End
FBM	Factional Brownian Motion
FECN	Forward Explicit Congestion Notification
FIFO	First In First Out
FPE	Final Prediction Error
FR	Frame Relay
GPS	Generalized Processor Sharing
IETF	Internet Engineering Task Force
IntServ	Integrated Services
LAN	Local Area Network
LBC	Leaky Bucket Controller
LDP	Label Distribution Protocol
LMI	Linear Matrix Inequality
LQR	Linear Quadratic Regulator
LSP	Label Switched Path
MBAC	Measurement Based Admission Control
MEX	Matlab Executable File
MIB	Management Information Base
MMP	Markov Modulated Process
MMPP	Markov Modulate Poisson Process

MPLS	Multi-Protocol Label Switching
NCCT	Network Computing and Control Technologies Research Laboratory
NCIT	National Capital Institute of Telecommunications
NE	Network element
NMS	Network Management System
NSP	Network Service Provider
OTB	Out-of-Box
QoS	Quality of Service
PCR	Peak Cell Rate
PDF	Probability Density Function
PE	Provider Edge
PEM	Prediction Error Method
PGPS	Packet by Packet GPS
PPVPN	Provider Provisioned Virtual Private Network
PVC	Permanent Virtual Circuit
PSD	Power Spectral Density
PQ	Priority Queue
RAA	Cisco Service Assurance Agent
RE	Reactive Estimator
RED	Random Early Detection
ROI	Return On Investment
RPEM	Recursive Prediction Error Method

RR	Round Robin
RSVP	Resource Reservation Protocol
SAC	Service Admission Control
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SP	Service Provider
TE	Traffic Engineering
TFRC	TCP Friendly Rate Control
VPN	Virtual Private Network
VRF	Virtual Routing and Forwarding
VCI	Virtual Channel Identifier
WAN	Wide Area Network
WFQ	Weighted Fair Queuing

Symbols

A list of symbols on a per-chapter basis is summarized. The common used symbols are listed in the Chapter where they are first introduced.

Chapter 2

k	sample instant of a discrete system
N	number of samples in a discrete system
$y(k)$	output of a control system
$r(k)$	input of a control system
$x(k)$	state variable of a control system

J_N	quadratic cost function
H_2	optimal control normal
H_∞	optimal control normal
c	link capability
W	window size
Q_{ref}	desired queue length at a router
Q_t	actual queue length at a router Packet Loss Probability

Chapter 3

$\mathcal{C}$	a number of statistical Quality of Services
$\mathcal{C}_j$	individual class of service
P	packet loss probability
P_j	individual packet loss probability

Chapter 4

$A[0,t)$	Packets arrived from all of the sources in the interval $[0,t]$
$\alpha(k)$	the stochastic increment process in a discrete time setting
$P_\alpha(n)$	distribution of variable $\alpha(k)$
$\mathbb{P}$	Probability
$q(k)$	stochastic process to denote the length of the queue
b	maximal finite buffer size
c	link capacity
$l(k)$	instantaneous packet loss experienced by the multiplexer during each period $[k, k+1)$

$\hat{l}(k)$	measured packet loss in time interval $[k, k+1)$
$\hat{\alpha}(k)$	measured packet rate in time interval $[k, k+1)$
P_{loss}	packet loss probability
$\mathbb{E}$	mathematical expectation
plr	packet loss rate
$P_{overflow}$	buffer overflow probability
$I(x)$	identification function
$eb(\theta, t)$	effective bandwidth
θ	space parameter
t	time parameter
μ	mean of the arrived packets for each interval
$Z(t)$	normally distributed stochastic process with zero mean
H	Hurst parameter
σ^2	variance of the random variable
ϵ	estimation error
δ	estimation error range
ζ	small real number
S	a set of real numbers
inf	minimization function
sup	maximization function
N_l	total number of lost packets
N_o	total number of transported packets

Chapter 5

$w(l)$	linear prediction coefficients
Q	covariance matrix
H	constand gain matrix
e	residual
$\hat{X}_k^-$	apriori estimate
$\hat{X}_k$	aposteriori estimate
K	Kalman gain
P_k^-	apriori covariance
P_k	aposteriori covariance
p	window size
$r_{ab}(k)$	customer's packet rate of VPN connection ab
τ_{ai}	time delay along the path from the node a] to node i
T	time slot
$R_{ab}(k)$	calculated packet rates for VPN connection ab
K_p	P controller parameter
K_i	PI controller parameter
$D(z)$	digital controller
$G(z)$	transfer function

Chapter 6

M^*	a set of model
θ	identified parameter
Z	measured data set
$\hat{L}(k\|\theta^*)$	best one-step ahead output prediction
$\epsilon(k,\theta^*)$	one-step ahead output prediction error
$V(\theta)$	value norm function
$V'(\theta)$	gradient of $V(\theta)$]
$H(\theta)$	Hessian matrix
$\phi(k,\theta)$	gradient of the error
$\mathcal{V}$	loss function
S	sample rate of the sFlow
$I(x)$	identification function
α_0	identified parameter in the dynamic model
α_1	identified parameter in the dynamic model
β_0	identified parameter in the dynamic model
β_0	identified parameter in the dynamic model
λ	forgetting factor
γ	constant gain parameter

Chapter 7

$C(z)$	output of a control system
$D(z)$	digital controller of a control system
$R(z)$	input of a control system
$H(z)$	sensor of a control system
$G(z)$	transfer function of a control system
ω	natural frequency
ζ	damping ratio
A_f	closed-loop system matrix
λ	roots of the characteristic equation

Chapter 8

δA	uncertain parameter matrix of the identified system
δC	uncertain parameter matrix of the identified system
$L(X, \gamma)$	symmetric block matrix
$Re(\alpha)$	real part of the complex number α
$\lambda_i(A)$	i^{th} eigenvalue of matrix A
P	variable matrix
J_{min}	minimum quadratic cost
γ	specified upper bound
K	controller parameter $[K_p \;\; K_i]$

Chapter 1

Introduction

1.1 Motivation

The Internet, without a doubt, has changed our lives. A number of Network Service Provider (NSP) companies have emerged in order to make profits from the developing trend. Most of them continue to provide broader bandwidth connectivity to customers for the increasing demands of faster Internet access service. Unfortunately, the increase in traffic generated by the access service, which is a major traffic component in the IP network, has not been translated into a proportional increase in the NSP revenues. Therefore, NSPs must find other ways beside the access service to increase the profit margins while maximizing their large investments.

Meanwhile, the geographically distributed businesses ranging from small-sized to large corporations have to be connected together in order to efficiently utilize their Information Technology (IT) resources and applications. Since the number of remote users and branch offices will keep growing, these enterprises need a more affordable and scalable way to meet the connection demand. They also require all of these accesses to networked resources including legacy systems and new enterprise applications without compromising security and Quality of Service (QoS).

The *Virtual Private Network* (VPN) services [62] allow customers to connect several sites of their network over a provider owned network cloud, thus creating the appearance of a seamless organization network.

Originally, the solution involved the setup of leased line circuits between customer

end-points, which was effective and secure. Moreover, it provided deterministic QoS characteristics to the users. However, the main drawback was the higher cost: to use the VPN service, the customer had to reserve a full transmission line. This issue limited the number of customers that a network could support, hence the price was high.

To improve, the VPN service developed on packet switched networks, of which Asynchronous Transfer Mode (ATM) and Frame Relay (FR) based networks are the best examples. Both ATM and FR can provide the *end-to-end connectivity*, which simulates the circuit built through the leased lines. At the same time, the price is decreased by sharing network resources with other customers.

Internet Protocol (IP) has the lead as today's preferred choice due to lower cost, a higher adoption rate and advanced features. However, the IP network is connectionless by nature. The main service offered is the *access* service, which only provides communication guarantees as guaranteeing QoS is extremely difficult to implement in IP networks.

Fortunately, *Multiple Protocol Label Switching* (MPLS) [22] brings the end-to-end connectivity into the connectionless IP network. MPLS is an IETF specified framework that can support IP, ATM and FR protocols. As an application of MPLS technology, MPLS VPN [98] appeared as a solution to provide QoS enabled VPN services over the IP network.

MPLS VPN is becoming more and more popular, since it can satisfy both the demands of clients and service providers. It is more scalable and easier for clients to use, compared with the previously introduced layer 2 VPN (i.e. ATM or FR). For example, MPLS VPN can be smoothly upgraded from 10 Mbps to 100 Mbps just with software configuration. It is also much cheaper because it uses the largely deployed IP technology. On the other hand, the service provider can utilize the existing IP network infrastructure to get a higher Return on Investment (ROI) through provisioning an MPLS VPN service.

The MPLS VPN services bring a lot of market potential for the service providers. Thus, how to efficiently deploy and manage the MPLS VPN service is a key problem that needs to be solved for service providers. Currently most of the work is addressing security issues. Although IETF is actively pursuing the VPN service category [52] as part

of the work done by the *Provider Provisioned Virtual Private Networks* (PPVPN) group, it attempts to provide connectivity among the distributed VPN sites with minimal or no guarantees of QoS.

The growing dependence and use of real time, interactive and other QoS sensitive applications demand that effective QoS capabilities should also be included in the technology. The MPLS VPN is a technologically advanced network designed to support emerging IP applications such as Voice over IP (VoIP) as well as legacy applications. VoIP requires support for low delay and low packet loss. This is because VoIP is a streaming service where retransmission is not feasible. In addition VoIP requires high service availability. The level of availability depends on whether the VoIP service is intended to be a primary line service, available after a power outage or a secondary line service where service availability is not as critical. More VPN services mean more profit, but by accepting more VPN connections than the capacity of the link allows for and thus violating the QoS agreements, NSPs will lose customers. The QoS trade-off will ultimately determine the maximal profits obtained by the service providers.

Therefore, how to control and guarantee the QoS for the provided VPN services is a critical issue to be addressed for the service providers in order to maximize their profits and keep themselves competitive in the market. The following section will firstly give a short literature review on the QoS control mechanisms, while the details are presented in Chapter 2.

1.2 A Brief Review

Addressing QoS issues in IP networks has been extensively studied, where two significant architectures are available to approach this challenge. The first technology is known as *Integrated Service* (IntServ) framework. It provides the end-to-end QoS, but the QoS is neither scalable nor applicable for deployment in a real network, because each flow or connection's state information is required in the intermediate routers. The second architecture is *Different Service* (DiffServ) framework, which differentiates the services into several classes. No state information is needed, so DiffServ is scalable. Unfortunately, the DiffServ framework can only provide *qualitative* QoS and no *quantitative* QoS can

be guaranteed.

MPLS technology provides the traffic engineering foundation for network based application services, giving the end-to-end connection for the connectionless IP networks. Consequently the VPN traffic flow from one point to another is connection oriented and predictable, which makes the quantitative QoS guarantee possible in IP networks.

In the quantitative QoS model, several performance parameters are usually defined to describe the end-to-end QoS, such as packet transfer delay, bandwidth, packet loss probability and jitter. *Service Level Agreement* (SLA) [46] is used to contain sets of procedures and service targets agreed upon by customers and providers. While the bandwidth is easier to guarantee, maintaining the delay or loss metrics below a preset value presents great benefits, but poses a huge challenge. Packet delay is largely employed as the QoS parameter to be guaranteed in the *deterministic* QoS model, which usually results in over-provisioning of network resources and low network utilization. In the statistical QoS model, the packet loss probability is the premier QoS parameter, where the network allows some packet loss to occur and thus is more applicable for a practical application.

QoS assurances are intrinsically linked to the problem of congestion control in the network. It can be safely stated that *solving the congestion control problem represents the key to providing QoS over packet switched networks.* With a control theoretical perspective, there are two primary mechanisms to control congestion and provide QoS: *open-loop system* and *closed-loop system.*

The *open-loop system* is based on resource reservation and is proactive by nature. A new service will be accepted only if enough resources exist in the network, and once a new VPN instance is admitted, the QoS guarantees are enforced via resource reservations. With the open-loop model, the network operator can provide QoS assurances to the users, as well as keeping the network operation in check. However, the service guarantees largely depend on user-specified traffic characterizations. As it is extremely difficult to characterize user traffic, only estimated number of an upper traffic envelope are taken into account. Providers can improve this characterization by observing the actual traffic and building a stochastic description of its behavior, termed as Measurement Based Admission Control (MBAC) system. It is helpful for the provider to increase the

network utilization beyond simple addition of user traffic requirements. But it may give results that are quite far from the preset loss targets, since user's traffic is unpredictable.

Closed-loop control strategies can dynamically control the network based on the output of the system. For example, explicit rate control mechanisms allow the control system to dynamically adjust the traffic rate at the ingress of the network. When there is congestion and the QoS agreement is broken, the control system will throttle the aggregate rate of the VPN traffic that will be inserted into the provider's network; therefore the congestion will be released fast and QoS will operate below a preset value in a long-term run. Such a system works in a feedback manner, so it may provide better results. However, the dynamic control mechanism makes the system more complex and its applicability in the IP/MPLS network is not addressed in the literature.

In this thesis, the QoS control issues for MPLS VPN services from both the open-loop and closed-loop control point of view, will be studied. The transducer, dynamic model and various controller techniques are investigated and designed to verify whether or not a dynamic scheme can work well for an IP/MPLS network. As starting point, the packet loss probability (PLP) is firstly estimated based on the large deviation theory and infinite buffer model. As in reality the buffer size is limited and small it is expected that estimation formulae are not accurate enough for a feedback control of PLP to be performed. Two enhanced on-line technologies are developed to overcome these deviations.

The reason to use PLP for packet loss control is that the packet loss probability is slower than the dynamic buffer occupancy and thus has *inertia* which makes it a good candidate for a controlled parameter of a closed loop system. The packet loss probability model is identified by applying the identification theory on the real measured data. Again, the model has some uncertainties, either because of the linearization process or of the inaccuracy of the traffic model or both. To compensate for all inaccuracies and make the control system work in a practical network, the robust and optimal controller is designed to solve the cover for all uncertainties in the models used.

1.3 Contributions

The thesis's idea is to estimate, identify and control the packet loss probability for the MPLS VPN services in the core network. The packet loss probability is estimated with Large Buffer based Estimator (LBE) and Aggregate Traffic based Estimator (ATE). Because the estimation model is based on the linearization of the complex model and the assumption of large buffer size, the estimation result indicates the performance can be improved. Then one step forward Kalman filter and the Reactive Estimator (RE) are applied to optimally estimate the packet loss probability. The packet loss probability is slower and it can be controller with the close-loop system. Therefore, the packet loss probability system is identified first, and then controlled with classical, state variable, and best controller.

The focus of this thesis is to address the *quantitative* QoS control issues in a *statistical* model that arise from the provisioning and management of QoS enabled VPN services across the core IP/MPLS networks. The major contributions are summarized as follows:

- The QoS control framework for the VPN services is proposed and implemented, which combines service admission control and dynamic packet loss control together. Distributed computing technology is applied to implement the system as proof of concept [120, 114].

- In the packet loss controlling task, one of the key issues is to accurately estimate the packet loss probability based on the knowledge of the input process. Two estimation formulae are derived by applying the Large Deviation Theory (LDT) on the buffer overflow probability. A series of experiments are designed on the live NCIT*net2 network to evaluate the performance of the estimators under different traffic arrivals and different buffer sizes [118, 127].

- Because the estimators are based on the large-buffer assumption, while the buffer size is small in the practical network, the performance indicates that there is enough room for improving the accuracy of the estimators, after analyzing the estimation errors. A simple Reactive Estimator (RE) is constructed, which surpasses the original estimator with the capacity of adapting itself to the variable

contexts. Specifically, RE will automatically adjust itself by employing one dynamic item based on the feedback of loss ratio measurement; it can estimate the packet loss probability more accurately with the time-varying traffic model [117]. Kalman filter is a general method for the optimal estimation of a noisy measurement, using the estimation error obtained from the past measurement to fix the one-step prediction. A Kalman filter is applied to optimally recursive estimate the traffic mean and variance [125]. A series of experiments are devised to evaluate the performance of the new estimator and the comparison shows significant improvements [129].

- As one of the RE' applications, the hybrid centralized and distributed service admission control for VPN services is proposed and evaluated on the live network. Specifically, the distributed measurement system employs an intelligent agent-based architecture to measure the VPN traffic on all of the heterogeneous remote nodes. The packet loss probability is then estimated on-line to decide whether or not the new services are acceptable [120].

- While RE is used as the online transducer, the controller is designed for the analytical loss model. The results show that it is possible to control the QoS parameters with the dynamic control mechanism [124]. However, since the analytical model is simply linearized to describe the real model, the controller cannot control the system well and the steady error is also significant.

- To improve the performance, the modeling of the dynamic plant is one of the key issues needing attention. System identification theory is applied on the dynamic packet loss system to identify the model, where the ingress traffic rates are inputs and the packet loss probability is the output. Both of the offline and online Prediction Error Methods (PEM) are explored [126, 130].

- Controlling the identified model is examined by both classical and modern controller design technologies. The digital controllers dynamically throttle the VPN ingress traffic to maintain the loss experienced at the core network below a preset target. Through a number of experiments, the transient and steady state performance is evaluated [121, 115].

- Since the identified linear model contains time-varying and norm-bounded uncertain parameters, a Linear Matrix Inequality (LMI) approach for designing the robust and optimal controller is studied. The robust and optimal control design problem is formulated as LMIs. By adopting the LMI expression, a symmetric positive definite matrix P with guaranteed overall system's quadratic stability can be easily determined. The matrix P can finally be used to infer the controller parameters [122, 123, 116].

- The main results in this research are assembled in a complete closed loop control system. The feedback control of the packet loss probability in MPLS VPNs uses a Kalman filter for the estimation of the packet loss probability, a robust controller to keep the probability of packet loss in desired limits and an on-line identification mechanism for conducting the controller poles in an adaptive mananer [128].

In summary, a framework for QoS parameter control, specifically the packet loss probability control, with explicit application to VPN services is addressed.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2 is the literature review, which includes the background, concepts, related techniques and comments for the QoS control and guarantees in packet switched networks.

Chapter 3 gives the description of the whole QoS control architecture and its implementation proposal. The following chapters implement each functional module in the control system.

Chapter 4 presents the details of the methods to measure and estimate the packet loss probability, and these estimators are evaluated in the live network. The online estimator is then constructed in Chapter 5, where it is applied to the service admission control and directly feedback service control. Chapter 6 proposes the identification theory and method to identify the dynamic packet loss system. Based on the identified model, the digital controller is designed in Chapter 7 to control the preset QoS parameters. Since

the identified model has uncertain parameters, Chapter 8 gives the investigation on the robust and optimal control technology for models with uncertainty.

In Chapter 9 the main results of the research developed by the author of this thesis are assembled in a complete closed loop control system. The feedback control of the probability of packet loss in MPLS VPNs uses a Kalman filter for the estimation of the probability of the packet loss, a robust controller to keep the probability of packet loss in desired limits and an on-line identificaiton mechanism for conducting the controller poles in an adaptive manner.

Finally, Chapter 10 concludes this thesis and proposes future research plans.

Chapter 2

Literature Review

This chapter serves as a literature review for the schemes of congestion control and QoS assurance in packet switched networks. The required background information is provided and relevant works are cited and commented. The chapter ends with a short section discussing the candidate approaches for the unresolved issues.

2.1 Quality of Service Overview

2.1.1 The Nature of QoS Issues

Quality of Service (QoS) refers to the ability of the network to handle the traffic such that it meets the performance metrics of certain applications [6]. It also means the network's ability to provide better services to some selected network traffic. QoS assurances are intrinsically linked to the problem of congestion control in the packet switched network. It can be safely stated that solving the congestion control problem represents the key to providing QoS over packet switched networks.

In a circuit switched communication network, there is a dedicated communication path between two end stations. The path is a connected sequence of links between the network nodes. On each physical link, a channel is dedicated to the connection. Once the line is setup, there is no resource sharing, and QoS is hard guaranteed. However, in a packet switched network, because resources (such as link bandwidth and memory buffer) of a link are shared among all the nodes attached to it, network congestion [106] will appear as a resource sharing problem. Specifically, a packet switched network

can be looked at as a network of queues. For each queue, all of user requests for data transmission need to be served, which are often unpredictable and bursty with regard to transmission starting time, traffic rate and packet size. Since any physical resource in the network has a finite capacity, network congestions will result when the resources cannot meet all of the users' current demands. Adverse effects from such congestions include the long delay and packet loss of message delivery, waste of system resources, and possible network collapse, which implies that the QoS cannot be guaranteed for the customers.

Adding more memory in the switch nodes may help up to a point to absorb the instant bursty data, but the problem of congestion cannot be solved by introducing infinite buffer space inside the network [81]. Firstly, in such a case the queues would then grow without bounds, and the end-to-end delay would increase. When the packet lifetime is finite, packets coming out of the congested router would have timed out already and need to be retransmitted by the transport protocols. In fact, too much buffer space in the switched nodes can be more harmful than too little, because the packets will have to be dropped only after they have consumed valuable network resources. Secondly, in contrast to Poisson traffic models, linear increases in buffer size do not result in large decreases in packet drop rates, because the traffic rates do not simply rise to a level, stay there for a while, and then subside. Periods of traffic congestion can be quite long with losses that are heavily concentrated [90]. This understanding of the behavior of congested networks suggests that it would be difficult to efficiently allocate the buffer size to reduce congestion adequately because the level of busy period traffic is not predictable.

2.1.2 QoS Metrics

To discuss QoS, it is important to first understand the performance metrics and how they are measured. The QoS metrics defined by *Service Level Agreement* (SLA) [46] usually include bandwidth, loss, delay and delay variation (also called jitter). These parameters are explained as follows:

Delay

The total delay experienced by packets can be composed of four delay categories: processing delay, transmission delay, propagation delay and queueing delay [103]. There are two latency metrics to measure the packet delay: *one-way* delay and *round-trip* delay.

The one-way delay [3] is the amount of time it takes a packet to reach the destination after its transmission from the source. It is usually measured by sending probe packets with timestamps between measurement points. However, after lots of studies in the literature [91], a number of fundamental obstacles still exist, which have been pointed by Papagiannaki [87] as the followings:

- Firstly, packet timestamps must be accurate enough to allow the calculation of the delay through a link or a path. This accuracy demand requires in particular that the measurement systems can offer sufficient resolution to distinguish the arrival times of two consecutive packets. Furthermore, the measurement system must be synchronized to an accurate global clock signal, such as *Global Positioning System.* These two conditions must be met in order that the maximum clock skew between any two measurement stations is limited enough to accurately calculate the delay of a packet from one point to another in the network.

- Secondly, the amount of data easily reaches hundreds of gigabytes in the current high-speed network, since data from ingress and outgress links need to be matched in order to compute the time spent in the link or path.

Both of the problems make the one-way delay measurement unrealistic in an operational network, since the calculations usually require expensive and sophisticated test gears and are beyond the budget and expertise of most enterprise customers.

The round-trip delay [4] is the amount of time it takes for a packet to return to the transmitter after it is echoed by the destination. Without the needs of clock synchronization, measuring the round-trip delay is easier and requires less expensive equipments. In reality, a general measurement of the one-way delay is usually calculated by measuring the round-trip delay and dividing it by two. For example, this method is used by Cisco Service Assurance Agent (RAA). However, since packets may take different paths in each direction, the one-way delay in each direction is possibly different and

that situation is not uncommon in the current high-speed network. If the NSPs want to provide quantitative delay guarantees, it is still necessary to measure the one-way delay independently, which results that the packet delay is not easy to be used as a QoS metric in the operational network.

Jitter

Jitter measures the variability of delay of packets in the given stream, which is an important property for many applications such as the media stream. For example, if one packet requires 100 milliseconds (ms) to traverse the network from one point to another, and the following packet requires 102 ms to make the same trip, then the delay variation is calculated as 2 ms.

Unavoidable jitter is caused by the variable queuing and propagation delays. Jitter is quantified in two ways: one is called *delay jitter*, which bounds the maximum difference in the total delay of different packets. This approach is useful for interactive communication, since a guarantee on the delay jitter can be translated into the maximum buffer size needed at the destination. The other one is called *rate jitter*, which bounds the difference in packet delivery rates at various times. That is, the rate jitter measures the difference between the minimal and maximal inter-arrival times.

The measurement of jitter is usually approached by comparing the one-way delay of each packet. With the same reason for the one-way delay measurement, it is hard to be implemented in the operational high-speed network. Moreover, jitter is mainly used to evaluate the QoS for *Voice over IP* (VoIP), since if the jitter is large, the call quality is greatly degraded. This thesis only deals with the VPN services, so both of the parameters, delay and jitter, are not very appropriate to be employed as the QoS parameters in this context.

Loss

In the live network, the multiplexer on each network element has a finite buffer. If packets arrive too fast for the node to process them, then some of the packets will be dropped or lost.

The methods for measuring packet loss can be classified into two categories [9]. The

first one is called *active* method, which usually uses a probe tool such as *ping* to send a series of packets aimed at a target system and measure the returned response packets. Unfortunately, the sampling nature limits the resolution and can also skew the results if the probing rate is too high [9]. The second one is named *passive* method, where passive monitors are attached to links or network elements. For example, the set of *Management Information Base* (MIB) counters can be retrieved through the *Simple Network Management Protocol* (SNMP) from the routers or switches. These counters record the number of lost packets and thus can be used to calculate the packet loss during each defined period.

However, the packet loss experienced by the multiplexer during each period is instantaneous and has a higher variation. Therefore it is not a very useful QoS metrics. A more appropriate variable is named in terms of the overall *Packet Loss Probability*, which is defined as the ratio between the expectations of packet loss and the arrival rate.

The packet loss probability can be calculated through measuring and averaging the long term packet loss and total amount of traffic. But if the packet loss probability is very low, such as 10^{-5} that is not uncommon in the high-speed network, the required time to get an accurate measurement will be too long and unacceptable in the operational network.

Because directly measuring packet loss probability is inapplicable in the high-speed network, some studies have focused on the estimation of packet loss probability from the *probability of overflow* that can be calculated with the stochastic characterization of input traffic. In such a scheme, the multiplexer is modeled as a theoretical system with an infinite buffer space. All ingress traffic is eventually sent on the egress link in the infinite buffer system, while the finite buffer system experiences actual loss. Although loss does not actually occur in such a theoretical model, it is assumed that packets encountering a queue that is longer than a fixed value are of no use and are considered as lost by the application. The probability of overflow is then defined as the probability that the queue size is longer than a given fixed capacity.

The packet loss probability will be employed as the primary QoS parameter in this thesis, and its measurement and estimation from the overflow probability will be ana-

lyzed in detail in Chapter 3.

2.1.3 QoS Architectures

During the past several years, numerous mechanisms have surfaced for providing QoS for communication networks. The fundamental objective of any QoS mechanism is to ensure that excessive congestion does not occur for the packets with assured QoS. It is important to keep in mind that QoS mechanisms do not create additional capacity, but only support prioritization of traffic and allocation of capacity under congested conditions, or reduce the source rates to decrease congestion. Over-provisioning and traffic engineering are other QoS mechanisms that strive to provide the same level of performance for all packets by avoiding congestion. For example, over-provisioning can be achieved by monitoring the flows and providing a capacity for twice the maximum flow size, so that the network never runs above 50 percent utilization. Traffic engineering also monitors the traffic flows in the network, but it then applies some algorithms to route the traffic into an uncongested path. Although they are also very important, these mechanisms are not reviewed in this thesis.

Service Models

The Internet Engineering Task Force (IETF) has proposed many service models and mechanisms to meet the demand for QoS. *Integrated Service* (IntServ) and *Differentiated Services* (DiffServ) models are the most notable.

IntServ, along with the *Resource Reservation Protocol* (RSVP), can provide end-to-end service guarantees in connectionless IP networks. Two additional services are proposed in this model: *Guaranteed Service* [102] and *Controlled Load Service* [113]. The first service can be used for the real time applications with strict bandwidth and latency requirements. With guaranteed service, no packets should be lost due to buffer overflow and there must be a specified upper bound on the queuing delay through the network. The latter service can support the traditional applications that want the network performance to be equivalent to a lightly loaded best-effort network. That is, the controlled load service ensures that a very high percentage of the packets do not experience delays that greatly exceed the minimum transit delay. However there is no

specified upper bound on the queuing delay through the network.

IntServ are implemented by four components: the signaling protocol, the admission control routine, the classifier and the packet scheduler. Applications requiring guaranteed or controlled load service must setup 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 classification and put the packet in a specific queue based on the classification result. Then the packet scheduler will schedule the packet accordingly to meet its QoS requirements.

The main problem with the IntServ architecture is the scalability issue. Since the routers need to keep per flow information, the amount of state information increases proportionally with the number of flows and this places a huge storage and processing overhead on the routers.

Because of the difficulty in implementing and deploying integrated services, DiffServ model is introduced, which aims to simplify core router complexity. The Differentiated Services architecture [15] is designed to provide differing levels of QoS to different traffic flows.

By marking the DiffServ fields of packets and handling them differently, several differentiated service classes can be supported. That is, differentiated service is essentially a qualitative QoS scheme. The following services are defined [58]: *Premium Service*, which is for applications requiring low delay and low jitter service. *Assured Service*, which is for applications requiring better reliability than best-effort service. *Olympic Service*, which provides three tiers of services: gold, silver, and bronze with decreasing quality.

Such a scheme is more scalable since the amount of state information is proportional to the number of classes and the supporting data structures rather than the number of flows. Further, the classification, marking and policing operations are only needed at the boundary of the networks.

MPLS is originally a forwarding scheme, which evolved from tag switching. In MPLS, a traffic trunk is an aggregation of flows with the same service class that can be put into a label-switching path. It provides a faster packet classification and forwarding

mechanism. More importantly, it provides an efficient tunneling mechanism for the connectionless IP network. That is to say, it creates end-to-end connection for the connectionless IP network.

MPLS can be used together with differentiated services to provide QoS. The only difference is that MPLS QoS is based on the Class of Service (CoS) bits in the MPLS label, whereas IP QoS is based on the IP precedence field in IP header. It also can provide the fine-grained guaranteed services using the RSVP protocol [133].

Quantitative Vs. Qualitative

Although many QoS architectures and algorithms have been proposed and developed, only two basic services are defined in the perspective of how to measure the QoS: *quantitative* QoS and *qualitative* QoS. The provision and guarantees of each service are appropriate for different applications.

Qualitative QoS is the ability of networks to provide better services to some selected network traffic, such as gold, silver and bronze services with decreasing quality. While *quantitative* QoS provides some QoS parameter guarantees for different type of services, for example one guaranteed service with end-to-end delay bound of 5ms.

The subject of providing quantitative guarantees in packet-switched networks has been a major area of research over the past 10-20 years or more, which firstly results in the specification of guaranteed QoS [102]. With this service, the packets have guaranteed delay bounds and no packet loss. Most of the studies in this field are based on the theory of network calculus [31] and [32].

Deterministic Vs. Stochastic

The *deterministic* QoS framework usually provides a hard exact bound on the QoS parameters, especially the explicit packet delay bounds. It has a number of limitations that restrict its applicability. Firstly, while there exists good closed-form results for systems that separate flows inside the network [88, 89], if traffic from several sources is aggregated inside the network, no deterministic guarantees can be made [32]. This is a result of the lack of information about what happens to the individual streams inside the aggregates. Secondly, the flow setup procedure cannot be automated, since there is no

general solution to the problem of proving delay bounds in general network topologies [73]. Finally, providing exact (deterministic) bounds on the delay means guaranteeing no loss at the same time. Obviously, this approach is only of theoretical interest, no network provider wants to offer such a service.

The above problem can be circumvented by adopting a *stochastic* modeling perspective. To provide such statistical QoS, the network can tolerate a small rate of packet loss. Hence it is more practical in the operational network. Obviously, packet loss probability is the appropriate QoS parameter in the framework for guaranteeing statistical QoS. Reisslein [95] also proposed a framework where the bound on the fraction of traffic that exceeds an end-to-end delay constraint is employed as the statistical QoS parameter.

Since the statistic QoS model is more appropriate for the VPN services, the packet loss probability parameter is used to define the SLA of the VPN services and tried to be maintained in this thesis.

2.1.4 QoS Approaches from a Control Perspective

QoS is mainly assured by traffic and resources management whose role is to prevent the network from being congested. Therefore, QoS assurances are intrinsically linked to the problem of congestion control in the network. In order to maintain a network in a healthy working condition and provide the performance guarantees, certain mechanisms have to be introduced to prevent the network from operating in the congested region for any significant period of time.

The main focus of this thesis is on the quantitative aspects of the VPN service, as defined by their QoS characteristics, rather than the qualitative service characteristics. In such a scheme, the QoS control system tries to maintain the QoS parameters below some preset values. On the other hand, a data network is an interconnected collection of autonomous computers, such that the network can be modeled as a distributed dynamic system. Therefore, the QoS control can be viewed from a control theoretic perspective [47]. The aim of the control system is to dynamically maintain the guaranteed parameter within certain normal levels (the packet loss probability in the present thesis context), so that the performance meets the QoS demands.

Many control algorithms have been proposed and developed in the fields. From a

control theoretical viewpoint, there are two basically control models: *open-loop control system* and *closed-loop control system.*

Open-loop control is an approach that the control decisions do not depend on any sort of feedback information from the system output. It is simple to implement, but if congestion occurs, the system does not have the ability to dynamically take some correct actions, such that QoS assurance may be violated. Closed-loop control system uses the feedback signal to dynamically adjust the system behavior. Thus it can control the congestion and maintain the preset QoS target in a better way. However, the closed-loop system will be much more complex. Due to the large delay and the statistic characteristic of the traffic, its stability and applicability are not investigated in the IP network.

The control technology and related system theory will be reviewed and introduced in the next section.

2.2 System Identification and Control

This section reviews system identification and control technologies applied to the QoS Control.

2.2.1 System Identification

A system is viewed as a process in which input signals are transformed by the system, resulting in other signals as outputs. The systems respond to particular signals by producing other signals or some desired behavior. Physical system in the broadest sense is an interconnection of components, devices, or subsystems.

To analyze and control a dynamic system, mathematical models are usually employed to describe the relationship among certain variables of the system. There are basically two approaches to the problem of building a mathematical model of a given system [76].

1. The first procedure is called direct modeling. Based on the physical laws and relationships that govern the system's behavior, a mathematical model can be obtained. In this procedure, it is required to look into the mechanisms that generate signals and variables inside the system.

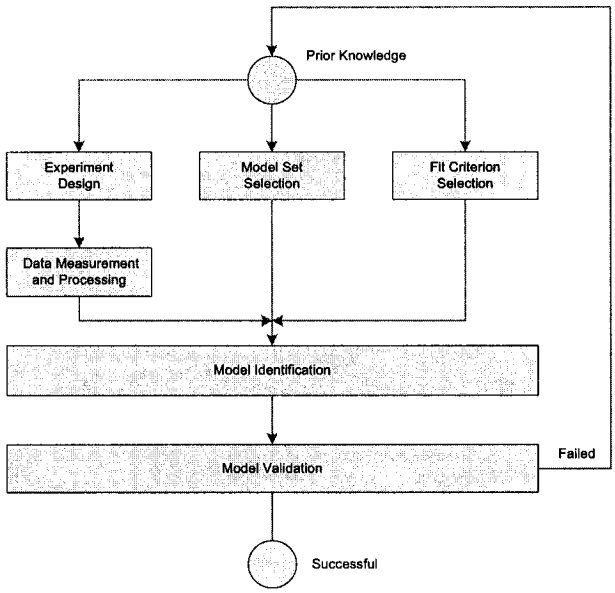


Figure 2.1: System Identification Procedure

2. Sometimes, it is impossible for direct modeling; for example, the system may have incomplete mechanisms or unpredictable properties. Moreover, some derived models are unnecessarily complex. In such cases, an experimental approach is used to construct the model based on signal measurements from the system. The approach is known as the *System Identification* method.

In practice, system identification method usually builds mathematical models of dynamic systems based on both prior system knowledge and observed experimental data, and is thus a combined theoretical and experimental modeling method.

Essentially, system is identified by adjusting parameters within a given model until its output coincides as well as possible with the measured output. The procedure of system identification has a natural logical flow as shown in Figure 2.1. The process repeatedly select a model structure, compute the best model in the structure, and evaluate this model' properties to see if they are satisfactory. The cycle can be itemized as follows:

1. Design an experiment and collect input-output data from the process to be identified

2. Examine the data. Polish it so as to remove trends and outliers, select useful

portions of the original data, and apply filtering to enhance important frequency ranges.

3. Select and define a model structure (a set of candidate system descriptions) within which a model is to be found.

4. Compute the best model in the model structure according to the input-output data and a given criterion of fit.

5. Examine the obtained model's properties.

6. If the model is good enough, then stop; otherwise go back to Step 3 to try another model set. Possibly also try other estimation methods (Step 4) or work further on the input-output data (Step 1 and 2)

System identification techniques apply to very general models. Most common models are difference equations descriptions, such as Auto-Regressive (AR) and auto-Regressive Moving Average (ARMA) models, as well as all types of linear state-space models.

Any estimated model, no matter how good it looks on the screen, has only picked up a simple reflection of reality. Surprisingly often, however, this is sufficient for rational decision making.

2.2.2 Control

In recent years, control technologies play an increasingly important role. Practically, every aspect of our day-to-day activities is affected by some type of control system. In the control system, the controller is placed within the system to modify the dynamics of the system so that a more satisfactory system response is obtained.

From a control theoretical viewpoint, there are two basically control models: *open-loop control system* and *closed-loop control system*, shown in Figure 2.2 (a) and (b), respectively.

The elements of an open-loop control system can usually be divided into two parts: the controller and the controlled process, as shown in Figure 2.2 (a). It is not difficult to see that the open-loop control system would not satisfactorily fulfill critical performance requirements. For instance, if the reference input is set at a certain initial value, which

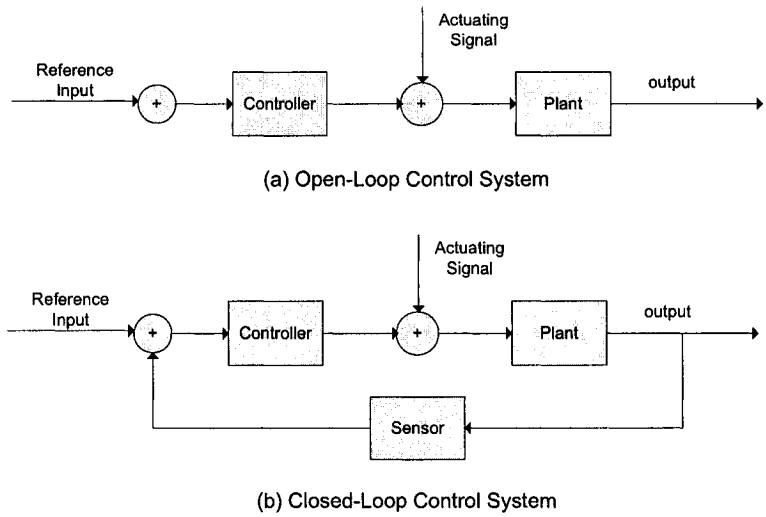


Figure 2.2: Block Diagram of a Basic Control System

corresponds to a certain output, when a noise is then applied, there is no way to prevent the deviation of the output. Because of the simplicity and economy of open-loop control system, this type of system is usually found in many non-critical applications.

What is missing in the open-loop control system for more accurate and more adaptive control is a link or feedback from the output to the input of the system. To obtain more accurate control, the controlled signal should be fed back and compared with the reference input, and an actuating signal proportional to the difference must be sent through the system to correct the error. A system with one or more feedback paths is how closed-loop systems are structured.

System Performance

The motivation for using feedback is to reduce the error between the reference input and the system output. However, the effect of feedback in control system is more complex and other performance characteristics should be checked.

- **Stability**. Stability is a notion that describes whether the system will be able to follow the input. A system is said to be unstable if its output is out of control or increases without bound. Feedback can improve stability or be harmful to stability if it is not properly applied. Many techniques can analyze the stabilities,

such as Routh-Hurwitz criterion, root-locus procedures, and frequency-response methods [92].

- **Sensitivity**. Sensitivity is important in the design of control systems. Since all physical elements have properties that change with environment, the parameters of a control system is not completely stationary over the entire operating life of the system. A good control system should be very insensitive to parameter variations but sensitive to the input.

- **Time Response**. Since time is used as an independent variable in most control systems, it is usually of interest to evaluate the state and output responses with respect to time. The time response of a control system is divided into two parts: the transient response and the steady-state response. Transient response is defined as the part of the time response that goes to zero as time becomes very large. All real control systems exhibit transient phenomena to some extent before the steady state is reached. Maximum overshoot, delay time, rise time and settling time are the most important performance criteria used to check the transient response when designing the controller. The steady-state response of a control system is also very important, since it indicates where the system output ends up at when time becomes large. If the steady-state response of the output does not agree with the steady-state of the input exactly, the system is said to have a steady-state error. Steady-state error is another performance criterion to measure the system accuracy.

Digital Controller Design

The emphasis of the control system is to design a controller transfer function that will satisfy design specifications for a given system.

The classical design techniques are based on the system transfer function, such as frequency design method and root-locus method. The root locus for a system is a plot of the roots of the system's characteristic equation as gain is varied. The design procedure is to tune the loop gain in order to shift the roots of the characteristic equation to more appropriate locations in the z-plane.

The more modern approaches to the analysis of discrete-time systems employ the pole assignment techniques.

Optimal Control

The classical control method is effective, but is largely trial and error, while experience is very important. Even when an acceptable design is completed, the question remains as to whether a better design could be found. The pole assignment design technique is known as a modern technique, where the exact pole location can be specified for the closed loop equation. However, the assumption must be made that the exact pole locations that yield the best control system are known.

Optimal control technique is developed to yield the best control system. A mathematical function, termed the cost function, is assumed. The optimal design procedure minimizes this cost function, hence the system is optimal.

For discrete system, the cost function is generally formed as follows:

$$J_N = \sum_{k=0}^{N} L[y(k), r(k), u(k)] \tag{2.1}$$

where k is the sample instant and N is the terminal sample instant. The control system outputs are $y(k)$, the inputs are $r(k)$, and $u(k)$ are the control inputs to the plant.

For a physical system, the control inputs are always constrained, precisely expressed by

$$|u_i(k)| \leq U_i \tag{2.2}$$

where each U_i is a given constant. The availability of finite control energy may be represented by a term in the cost function as

$$\sum_{k=0}^{N} u^T(k) R(k) u(k) \tag{2.3}$$

where $R(k)$ is a weighting matrix and is not related to $r(k)$.

Usually a quadratic cost function of the states and the control signals is studied in the optimal control literature, since the development is simple and the cost function is logical in practice. The linear quadratic optimal control problem is stated as follows:

For a linear discrete time-invariant plant described by

$$x(k+1) = Ax(k) + Bu(k) \tag{2.4}$$

$$y(k+1) = Cx(k) \tag{2.5}$$

determine the control

$$u(k) = Kx(k) \tag{2.6}$$

that minimizes the quadratic cost function

$$J_N = \sum_{k=0}^{N} x^T(k)Qx(k) + u^T(k)Ru(k) \tag{2.7}$$

where N is finite, Q is positive semi-definite, and R is positive definite.

The design solution is solved as follows [92]:

$$P(N) = Q(N) \tag{2.8}$$

$$u(k) = -K(k)x(k) \tag{2.9}$$

$$K(k) = [B^T P(k+1)B + R]^{-1} B^T(k+1)A \tag{2.10}$$

$$P(k) = A^T P(k+1)[A - BK(k)] + Q \tag{2.11}$$

where $P(N) = Q$ and $K(N) = 0$.

Robust Control

Another issue is that the linear model may contain some uncertain parameters because of either time-varying characteristic or the linearization process. The controller should be designed to perform satisfactorily over the anticipated range of plant parameter variations.

The term *uncertainty* refers to the differences or errors between models and reality [134]. To be practical, the identified system model cannot exactly describe the physical system. The inevitable inadequacy exists in all of the models. In order to be applicable in the practice, the model is highly simplified; otherwise it would be too complicate to handle for a controller design procedure. Therefore, the controller design is based on a crude mathematical model of the physical reality.

There are two basic types of uncertainty, parameter uncertainty and unstructured uncertainties. Representing uncertainty is usually to separate the uncertainty from the plant by creating an additional block which is connected to the plant by feedback. The general uncertain feedback system is depicted in the following Figure 2.3

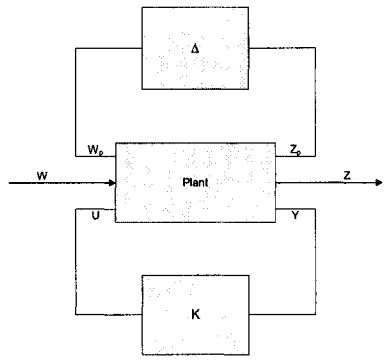


Figure 2.3: General Framework for an Uncertain Feedback System

H_2 and H_∞ norms are used to measure control system properties. A norm is an abstraction of the concept of length. Both of these techniques are frequency domain techniques. H_2 control seeks to bound the power gain of the system while H_∞ control seeks to bound the energy gain of the system. Gains in power or energy in the system indicate operation of the system near a pole in the transfer function. These situations are unstable. H_2 and H_∞ control are discussed in [134] .

2.3 Open-loop QoS Control

While closed-loop control is the general rule in packet switched networks, it was initially thought to be inappropriate for broadband *Wide Area Networks* (WAN) due to the high delay-bandwidth product of the control system. The delay bandwidth product determines the amount of data that can be in transit in the network. A high value brings considerable inertia in the realization of rate changes and leads to large buffer requirements [96, 72]. This led to the early definition of open-loop control strategies. In such a scheme, one must try to prevent the congestion to happen, i.e., it should be preventive-control in nature.

The open-loop control system is usually composed of admission control decision, traffic policing, resource reservation and packet scheduling as shown in Figure 2.4. The reference QoS parameters can be bandwidth, delay, jitter or packet loss, which will be controlled or guaranteed by the open-loop schemes. This model allows the network

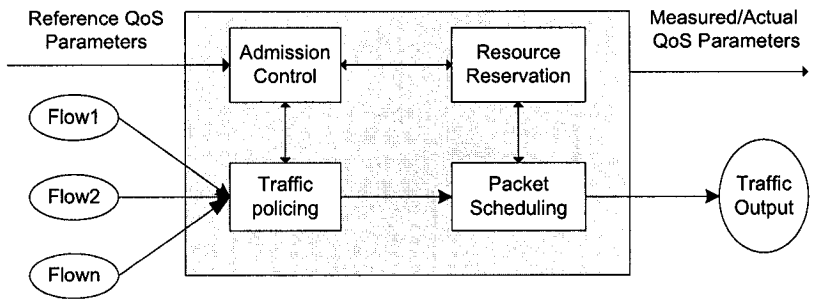


Figure 2.4: Block Diagram of an Open-loop QoS Parameter Control System

operator to make QoS assurances to users, as well as keep the network operation in tight check.

2.3.1 Admission Control

Admission Control (AC) is a set of mechanisms that accept or reject network traffic at either node or path level [96] to support the QoS demands via resource reservation. Conventional admission control systems, also known as *Connection Admission Control* (CAC), work on individual flows by applying the admission decisions to each flow.

Through admission control, the impact of a new flow on existing traffic is determined as *a priori* to the connection setup. Thus, a new flow will be instantiated only if enough resources are present in the network, and once a new instance is admitted, the QoS guarantees are enforced via reservations. Such a scheme is an integral part of ATM networks [51]. A similar type of scheme has now been developed for IP based Internet with RSVP protocol [133].

CAC is a very important controlling element and it is crucial to provide QoS guarantees under the *open-loop* model. Generally speaking, CAC needs three types of primary information for its successful operation:

1. An evaluation of the aggregate load (i.e. the existing traffic that has been admitted). This can be obtained from either the descriptions of the individual flows that form the aggregate or by estimating the aggregate directly. An admission control system that obtains the aggregate load information from the descriptions

of the admitted flows that have been provided by the user, will be called *Direct Admission Control* (DAC) or parameter based admission control, while if the aggregated load estimation is obtained from measurements, the resulting system will be called *Measurement Based Admission Control* (MBAC).

2. The traffic characteristic description of the new flow. The user is supposed to give some coarse descriptions of the amount of traffic that he plans to send, such as the peak rate and average rate, which are usually provided in the SLA.

3. QoS metrics and their preset values that were defined in the SLA and have to be assured for the flow by the service provider.

Direct Admission Control

DAC have been studied under both deterministic [88, 64, 31] and stochastic models [12, 104, 5]. However, the most of studies are focused on the deterministic QoS guarantees. For such a case, DAC algorithms are based on worst case bounds derived from the parameters describing the flow.

For example, it is widely used for IntServ networks to offer a bounded delay packet delivery service to support real-time applications. This type of service is called *Guaranteed Service* in the IntServ architecture. The work by Jamin [64] is the most notable study in this literature. His work is based on the theoretical results obtained in the deterministic analysis approach of Cruz [31, 32] and Parekh [88, 89]. They have proven that deterministic end-to-end delays can be calculated in networks where traffic is ingress bounded and PGPS schedulers are used in all nodes for individual flows.

Typically, the source is described by either peak and average rates or a filter like a token bucket, which provide upper bounds on the traffic that can be generated by the source. Then the algorithm computes the worst-case theoretical queuing delay to guarantee an absolute delay bound for all packets. Therefore, network utilization under this model is usually acceptable when flows are smooth; however, that typically results in low network utilization in the face of bursty network traffic.

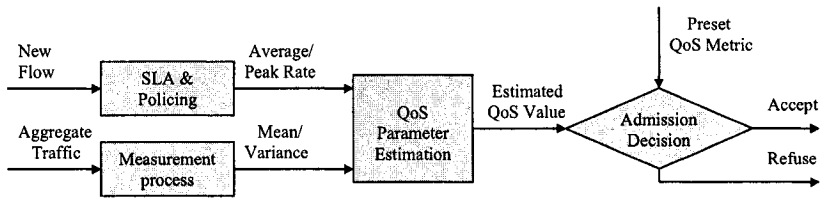


Figure 2.5: Block Diagram of a Measurement Based Admission Control System

Measurement Based Admission Control

DAC is also limited by the ability of the user to provide tight traffic characterizations, and the inability of the network to extract statistical multiplexing information from those descriptions. These problems can be circumvented by adopting MBAC. MBAC algorithm, which is less stringent, is employed for providing an enhanced quality of service without making hard guarantees [38, 14, 94, 101, 107, 29]. The admission control decisions are based on measurements of existing traffic rather than on worst-case bounds of traffic behaviors. Therefore, it can achieve much higher network utilizations while still providing some acceptable QoS.

In designing a measurement based admission control system, one must try to achieve the following two goals: one is to accurately estimate a priori the level of service failures and the other is to achieve the highest possible utilization for a given level of service failures. To reach both goals, two key components are required: a measurement process that produces an estimate of network load and a decision algorithm that uses this load estimate to make admission control decisions. The measurement process can be very different, which ranges from a simple point sample estimate to an exponentially weighted average. The decision algorithm uses a preset QoS parameter as the reference to decide if the additional flow can be accepted [64]. The block diagram is shown in Figure 2.5.

Since MBAC cannot provide a strict QoS guarantee, and the service must tolerate variance in packet delays and some packet loss, usually *Packet Loss Probability* is used as the main QoS target for the measurement based admission control schemes [96]. To guarantee the statistical QoS, the controlling system must try to maintain a value of the loss parameter below a preset number. Therefore, the main problem to be solved is to estimate the loss probability based on the input process. This problem has been

studied under various assumptions by many researchers. The seminal work of Anick et all. [5] studies the behavior of the FIFO scheduler when fed by a large number of on-off sources. Stern [104] extends Anick's work to more complicated Markov Modulated Rate Processes. The link to the notion of *effective bandwidth* is proposed by Kelly [66], [67]. Similar results for effective bandwidth approximation are also obtained by Guerin [55].

An estimator is proposed [35] for obtaining the effective bandwidth value from traffic measurements. Recently, Courcoubetis et al. [29] use the notion of traffic equivalence to speed up the numerical solution process. The method proposed is heuristic at this point, since the authors state that they could not find a proof for the convergence property of the proposed algorithm.

Qiu and Knightly [94] propose a system based on measurement of the *aggregate traffic envelope*. The envelope is a function returns the peak rate of the traffic over a given timescale. The purpose is to detect the time scale over which the maximum loss occurs. The idea of multi-time scale analysis first appears in the context of analyzing the multiplexing of multiple time-scale Markov streams [108]. The same line of reasoning is developed by Kim [68] as well as Montgomery [80].

Tse and Grossglauser [107] present a bufferless admission control framework that operates over several time scales. The time scale of interest is determined to be the flow lifetime. In contrast, for the virtual private network based services, the service lifetime can be assumed infinite, because each VPN service is composed of many flows. Nevertheless, variability in traffic can be observed, due to the aggregation at the customer to provider interface. Although time scale determination remains important, some other means have to be employed.

Another approach to MBAC is presented by Dziong [38], where a Kalman Filter framework is proposed to estimate the online mean and variance of the traffic. However, it is only designed for ATM networks. Gibbens and Kelly [50] introduce the Bayesian estimation as the means of determining if a flow should be admitted or not. The role of the measurement window is studied in the framework of single queue servers [107]. Grossglauser [54] and Duffield [36] also take into account the effect of sampling on the accuracy of the measurements.

Although the studies of stochastic modeling are heavily focused on the FIFO multi-

plexer, there are still a number of studies which have extended to the other multiplexers and networks, such as WFQ studied by Kumar et all [70] and static priority scheduler by Berger and Whitt [12]. In particular, Chang [23] obtains a large deviations result that is applicable to *in-tree* networks.

Breslau et al. have reported [21] that a number of MBAC algorithms seem to have identical performance, and infer from this that further work on this issue might not be required. However, first thing to be noticed is that the algorithms under consideration have different performance criteria, which are difficult to reconcile. In particular, the algorithm described by Crosby [30] and Qiu [94] use the loss rate as the tunable parameter, while the other algorithms use a utilization target. Secondly, it is not clear from how the utilization target can be used as a differentiator. The authors then plot the loss-utilization curves for the algorithms. Note that this can be misleading, because at a given load level for the same mixed traffic the *loss will be identical.* Plotting the loss-utilization curves does not tell anything about the algorithm performance. Furthermore, there is no time to measure and react when there is a large bandwidth or traffic speeds such as one or ten Gbps.

The previous studies are focused on the flow level, i.e., it is connection admission control. The case of virtual private services presents some crucial differences from this classical connection admission control problem exploited, as the followings:

- The rates of services setup and tear down are assumed to be rather small compared to the call setups that are experienced in phone networks for example (which was used as a base for simulations in the paper discussed above). This works to the advantage of a measurement based system because it allows it to acquire the necessary information before another request comes in.

- Traffic from a customer site is itself an aggregate of many sources, and as such presents different characteristics from the individual source traffic.

- Most of the network traffic presents strong temporal dependencies.

Since the call setup rate is linked to the speed of computation of the admission control procedure, a low setup rate of the VPN services implies that more computationally complex algorithms can be used, if the increase on robustness and consistence justifies

the computational overhead. It is important in this context for the algorithm to be consistent because usually there are penalties for breaching the QoS targets.

However, traffic measurements are not always good predictors of future behavior and so the measurement based approach to service admission control can lead to occasional packet losses that exceed desired levels. If such occasional service failures are controlled in the given range, the control system is acceptable given the relaxed nature of the service commitment provided by soft real-time services.

The main criterion used in evaluating any admission control algorithm must be how well it fulfills its primary role of ensuring that service commitments are not violated. The second evaluation criterion is how high a level of network utilization an admission control algorithm can achieve while still meeting its service commitments. These issues will be investigated in Section 2 of Chapter 5 to see if the service admission control scheme works.

2.3.2 Policing and Marking

To provide QoS assurance, each class of traffic must have certain limits to its allowable behavior such as how fast packets may arrive. These limits are usually referred to as a traffic profile, which is specified in the SLA when the contract is signed. *Policing* and *Marking* [6] are defined as the set of actions taken by the network nodes to control the user behavior with the purpose of maintaining user traffic within agreed values. Policing is a very strict policy that just simply drops the out-of-profile packets. However, the marking has a much softer touch that just marks the non-conforming traffic. The marked traffic serves as an indication to subsequent nodes on the path that these marked packets can be dropped first if congestion arises. With either or both of policing and marking, the network is protected from misappropriation of traffic resources, thus allowing the network control mechanisms to maintain the preset control targets.

Policing and marking share a common device that is to determine whether each packet is in or out of profile. The most popular device is the *leaky bucket controller* (LBC) [28]. This device forces traffic to conform with a linear, in case of the single leaky bucket, or concave, in case of the multiple leaky buckets, arrival curve. That is, a small degree of burstinees is allowed within a particular traffic class and the long-term

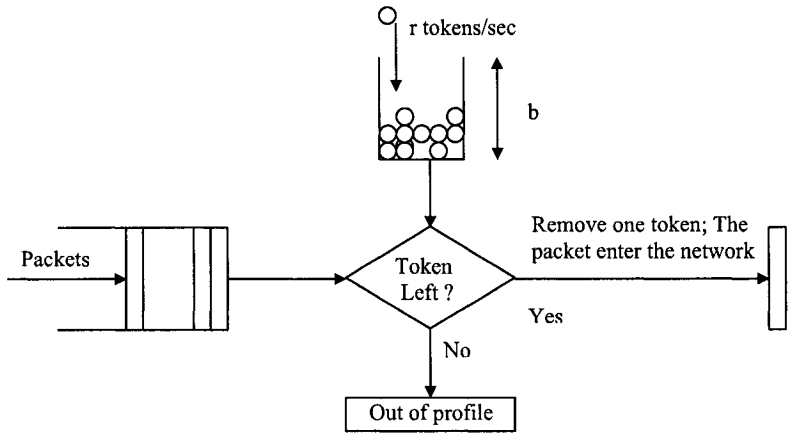


Figure 2.6: Block Diagram of a Token Bucket System

rate limit is enforced.

Each LBC is a dynamic system, that emulates a storage system of capacity b which fills at constant rate r and empties at the speed of the arriving traffic. The block diagram is shown in Figure 2.6. When a packet arrives and at least one token is available, the token is removed and the packet is considered to be in profile. If no tokens are in the bucket, that packet is declared to be out of profile. The token replenishment rate r is the long-term average rate limit for the arrival traffic. However, packets may arrive in short bursts and still be considered in profile. In such a case, there are up to b packets which may arrive back-to-back in time and still get through.

2.3.3 Packet Scheduling

Packet Scheduling dictates the characteristics of packet departures from each queue. It provides the NE with the feature of packet level control to support the QoS requirements. When there are network congestions, the resource allocation for a particular flow on each node is determined by the node's scheduling discipline, since the discipline decides which packet goes next from the queue and obviously, how often a flow's packets are served determines its allocated resources.

The models of all scheduling systems are similar: a set of buffers are drained by a server using a particular service discipline. Scheduling disciplines are classified in two

main categories: non-work conserving discipline and work conserving discipline:

Non-Work Conserving Discipline

The non-work conserving discipline does not necessarily send a packet even if one is present in the queues [132, 131]. This behavior is useful in maintaining inter-packet times within controlled limits, which in turn allows the control of the delay jitter. The drawback of the scheme is the lower link utilization and increased burden on the user to exactly describe its traffic. Therefore, the non-work conserving scheme is not supported too much by most of the operational networks.

Work Conserving Discipline

Work conserving disciplines always send a packet if one is queued in *any* of the queues. Almost all of the scheduler instances that are implemented in today's packet switching networks are of this type. Work conserving schedulers allow the user to make good utilization of the links through statistical multiplexing. However, they modify the characteristics of the individual flows, which make the prediction of the flow characteristic more difficult. The most popular disciplines are described and listed in the following:

- **FIFO.** The traditional packet scheduling mechanism on the Internet has been *First-In-First-Out* (FIFO) scheduling. With such a scheme, the packets are transmitted in the same order as they arrive in the queue. It is simple and easier to be implemented, which makes it the most popular in current operational networks. But FIFO cannot differentiate flows among the aggregated traffic. Thus it is hard to guarantee the specific performance bounds for a particular flow or provide prioritization among all of the flows.

- **PQ.** One of the most utilized scheduling disciplines is the *Priority Queue* (PQ) scheduler. PQ is extremely useful in the provisioning of a low latency traffic service. If the service is mapped into the highest priority queue at every hop, the worst-case latency only depends on the speed of the link and the maximum size of the packets. Its main advantages are given by the very good computational complexity, $\mathcal{O}(1)$, and the much simpler analysis under both deterministic and

stochastic settings. However, the drawback of this service discipline is that PQ allows higher priority queues to totally starve lower priority queues.

- **WFQ.** Another important class of work conserving service disciplines is represented by implementations of the *Generalized Processor Sharing (GPS)* policy and called *Weighted Fair Queuing* (WFQ) or *Packet based GPS* (PGPS). WFQ is the first packet scheduling algorithm with bandwidth and delay fair. It is proposed by Demers [34] and then analyzed by Parekh and Gallager [88] in a seminal paper. One important result proved by Parekh is that the difference between the idealized model (GPS) and the WFQ algorithm is bounded. This allows the analysis of GPS to be carried to the WFQ implementation.

  WFQ is significantly useful because it allows different weights to be applied to individual queues. It also has a good computational complexity, which is only $O(n)$ where n denotes the number of queues. The most important result regarding any packet based version of GPS policy can be summarized by Theorem 2.1, proved by Parekh in [88].

  Theorem 2.1 [88] The departure time of a packet under the PGPS link is bounded above by the sum of the departure time of the corresponding packet under the GPS link and the ratio between the maximum packet length (L_{max}) and the link capacity (c) ■

  The existence of a bounded difference between the GPS and the PGPS behavior allows the work under GPS assumptions with an easy shift toward the PGPS implementations that might be encountered in the network nodes.

- **RR.** An alternative algorithm, often termed Round Robin scheduling (RR), avoids local queue starvation by cycling through the queues one after the other and transmitting one packet before moving on to the next queue. The downside to this scheduling is that latency bounds are hard to enforce. Unlike the priority scheduling, it is not possible to assign one queue for low latency traffic, since every queue's service interval depends entirely on both the amount of other queues and the length of those packets in all of the queues at any instant.

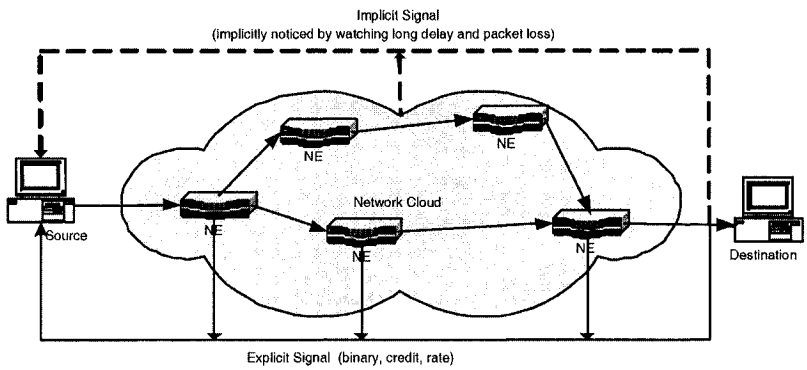


Figure 2.7: Mechanisms for Feedback Network Control

In the core network, however, only FIFO is deployed because of its simplicity in most of cases. Therefore, in this thesis, only FIFO is examined.

2.4 Feedback QoS Control

In the open-loop control the network operator makes QoS assurances to users, and keeps the network operation in tight check. However, due to un-predictive traffic, usually the network is forced to consider a lower utilization of the network in order to provide a better QoS guarantee. On the other hand, closed-loop control system can measure the QoS parameters and utilize those outputs as the feedback signals to dynamically adjust and control the system behavior, such that the network can fast restore to its normal state even after some unpredictable congestion has occurred. In such a way, it helps to avoid the QoS violation and maximize the utilization of the network resources.

From a control-theoretic viewpoint [47], the general depiction of a feedback control of network is provided in Figure 2.7. This object has been investigated in a lot of research works published in the domain literature. The basic issues are what to feedback and how the system to react. Several feedback signal schemes have been proposed [103]. General speaking, the signal can be *implicit* or *explicit.*

Using implicit signal is an effective congestion control technique in connectionless network. In such a case, the internal nodes do not send the explicit signal to the sources, and the sources know the system states just through the implicit feedback such as long

packet delay or large packet loss. For example, TCP mechanism [93] in IP network and LAPF [24] in FR network use this scheme.

With explicit congestion signaling, the network nodes alert end systems to growing congestion within the network, and the end systems take steps to reduce the offered load to the network. Typically, explicit congestion control techniques operate over connection-oriented networks and control the flow of packets over individual connections. The explicit congestion signals are classified into three general categories:

1. binary: A bit is set in a data packet as it is forwarded by the congested node. When a source receives a binary indication of congestion on a logical connection, it may reduce its traffic flow.

2. credit based: These schemes are based on providing an explicit credit to a source over a logical connection.

3. rate based: These schemes are based on providing an explicit data rate limit to the source over a logical connection. The source may transmit data at a rate up to this preset limit.

The control function acted by sources can be an *adaptive window*, in which the source indirectly controls the transmission rate by modifying the number of packets sent but not yet acknowledged (window), or an *adaptive rate*, in which the source, every time it sends a packet, sets a timer with a timeout value equal to the inverse of the appropriate transmission rate and transmits the next packet when the timer expires. The potential damage to the network is constrained in different ways, but window-based schemes are easier to implement because they do not require a fine-grained timer that is hard to implement in non-real-time operating systems.

The vast majority of the studies in the fields concentrate on specific network architecture, such as FR, ATM and TCP/IP. In the followings, these control mechanisms are reviewed according to the different network models.

2.4.1 Frame Relay Control Mechanism

The simplest way to cope with congestion for a Frame Relay network is to simply discard frames when congestion happens. Implicit signaling occurs when the network discards

a frame, and this fact is detected by the end user at a higher, end-to-end layer, such as LAPF control protocol. When this occurs, the end user software may deduce that congestion exists.

It is desirable to use much of the available capacity in a Frame Relay network as possible but still react to congestion in a controlled and fair manner. This is the purpose of explicit congestion avoidance techniques. In such a case, the network alerts end systems to growing congestion within the network and the end systems take steps to reduce the offered load to the network. Two general strategies were considered by Bergman [13]. One is called *Backward Explicit Congestion Notification* (BECN) and the other is *Forward Explicit Congestion Notification* (FECN). The choice of BECN or FECN may be determined at the configuration time. The frame handler may monitor the queue length and notify the users when the congestion is becoming serious. The user responds with adjusting window size, which must be done at a high layer.

Once congestion is detected, the protocol uses flow control to recover from the congestion. Let us assume the window size, W, can vary between the parameters W_{min} and W_{max}, and is initially set to W_{max}. In general, the window size W would like to be gradually reduced to throttle the transmission of frames as the congestion increases. Three classes of adaptive window decreasing schemes based on the response have been suggested by Chen [24].

1.1 Set $W = Max[W - 1, W_{min}]$

1.2 Set $W = W_{min}$

1.3 Set $W = Max[aW, W_{min}]$, where $0 < a < 1$

Successful transmissions (measured by receipt of acknowledgments) may indicate that the congestion has gone away and the window size should be increased. Three possible approaches are:

2.1 Set $W = Min[W + 1, W_{max}]$ after N consecutive successful transmissions

2.2 Set $W = Min[W + 1, W_{max}]$ after W consecutive successful transmissions

2.3 Set $W = Max[aW, W_{min}]$, where $0 < a < 1$

A study reported by Chen [24] suggests that the use of strategy 1.3 with $a = 0.5$ plus strategy 2.2 provides a good performance over a wide range of network parameters and traffic patterns; this is also the strategy recommended in LAPF. This window-based

scheme is simple and easier to be implemented. But the response time is long and its performance is low. Furthermore, it has the same problem with the TCP control mechanism presented in following section, that is the oscillating problem.

2.4.2 Available Bit Rate Service

In ATM network, the QoS is provided for *Const Bit Rate* (CBR) service, *Real-time Variable Bit Rate* (rt-VBR) service and *Non-real-time Variable Bit Rate* (nrt-VBR) service. However, the QoS is based on open-loop control approach and not suited to many data applications, such as file transfer that does not have well defined traffic characteristics. The Peak Cell Rate (PCR) by itself is not sufficient for the network to allocate resources effectively. Furthermore, such application can generally tolerate unpredictable delays and time varying throughput. ATM defines the *Available Bit Rate* (ABR) service to deal with this type of traffic. ATM Traffic Management Specification [28] provides the mechanism used for congestion control for ABR services. Since the nodes have computing ability, traffic and congestion control algorithms are fulfilled in ATM nodes and sources together. The control process is described as the following:

1. Every N_{rm} cells, the sources send a control cell
2. The network nodes measure load over a period
3. The network nodes specify a explicit rate and write into the control cell
4. The destination returns the control cell to the source
5. The source adjusts the transmission rate according to the explicit rate

Closed-loop control has been proved to be a promising mechanism for the ABR service category. New and more effective schemes are constantly devised for dynamically allocating available capacity in the network. Generally speaking, there are two ways to compute the explicit rates for each source. Firstly, it uses the current queue length and source rate to estimate the fair share rate for each source. Enhanced Proportional Rate Control Algorithm (EPRCA) [97] is in this type. The other way is to compute the fair rate allocation directly from a measurement of the available bandwidth and per connection state information. Explicit Rate Indication for Congestion Avoidance

Algorithm (ERICA) [65] is in this category. The latter way is implementation friendly (i.e., computationally inexpensive) and works well in a large number of situations, so that it appears to be favored by ATM switch designers.

The main drawback of these control mechanisms is that they do not provide any formal designing methodology to ensure the stability of the control system. Later on, a number of algorithms based on control theoretic principles appeared. A representative one among them is proposed by Kolarov [69], where an analytic method for the design of closed-loop congestion controllers is developed and the issues of stability and fairness are addressed.

All the control schemes for ABR services make effort to match the input rate to the available bandwidth. That is to say, it only can provide the bandwidth guarantees. However, because of the bursty traffic in data network, guaranteed-bandwidth-only policy can also result in huge packet loss and low QoS.

2.4.3 TCP Control Mechanism

In TCP/IP networks, best effort IP does not provide QoS guarantee mechanisms, so applications choose the preferable transport protocol to achieve the required performance. The current congestion control mechanisms, which detect and relieve congestion only at the endpoints, are called end-to-end congestion control, since it looks at the network as a black box. The credit-based flow control mechanism of TCP [93] was designed to enable a destination to restrict the flow of segments from a source to avoid buffer overflow at the destination. This same flow control mechanism is now used to provide congestion control over the Internet. The congestion control algorithm [63] used in the TCP protocol is a sliding window mechanism that uses packet loss to detect congestion. In particular, the end-systems probe the network state by gradually increasing the window of packets until the network becomes congested and drops packets. When a packet drop is detected, the window is shrunk until no losses occur. Then the window starts to increase until a packet drop is again detected. This control strategy obeys the principle that the network is a black box that cannot supply any explicit feedback information to the data sources. Thus, packet loss is the only implicit feedback signal to indicate congestion. Three major TCP mechanisms and implementations are: slow

start and congestion avoidance implemented by TCP Tahoe [63]; Fast retransmit and fast recovery implemented by TCP Reno [105, 2]; and TCP Vegas [20].

Jacobson [63] recommends a procedure known as *slow start.* This scheme probes the Internet to make sure that it is not sending too many segments into an already congested environment. The slow start algorithm has been found to work effectively for initializing a connection. He also proposes a linear growth in window size to deal with the congestion when a timeout occurs, which is called *congestion avoidance* and shown in the following steps:

1. Set a slow start threshold equal to half the current congestion window; that is, $ssthresh = cwnd/2$

2. Set $cwnd = 1$ and perform the slow start process until $cwnd = ssthresh$.

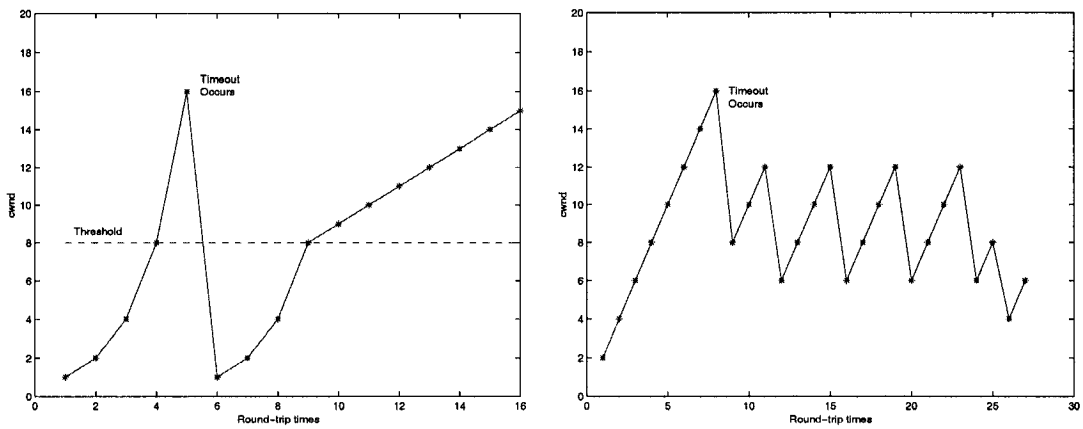
3. For $cwnd >= ssthresh$, increase cwnd by one for each round trip time.

If no more timeouts occur, the congestion window will continue to grow up to the size of the receiver's window. At that point, it will stop growing and remain constant as long as there are no more timeouts and the receiver's window does not change the size.

Based on the principle of *Additive Increase Multiplicative Decrease* (AIMD) [26], a TCP connection probes for extra bandwidth by increasing its congestion window linearly with time, and on detecting congestion, reducing its window multiplicatively by a factor of two.

One of the drawbacks is that today's TCP/IP is effective for pure best effort data transfer with little or no sensitivity to delay or packet loss, but makes problematic to support interactive applications. TCP is not well suited for several emerging applications including streaming and real time audio and video because its reliability and ordering semantics increases end-to-end delays and delay variations. Furthermore, many of these applications do not react well to the large and abrupt reductions in transmission rate on packet losses.

TCP-Friendly Rate Control (TFRC) [56] can smoothly adjust the transmission rate while coping with network congestion. However, it does not consider the control influence on the application level performance. For example, the TFRC sender changes its



TCP **AIMD**

Figure 2.8: Dynamic Window Sizing on Congestion

sending rate at least once per Round-Trip Time (RTT). Such a frequent rate control obviously affects the perceived video quality.

Model-based congestion control uses a TCP model instead of a TCP-like AIMD mechanism. By adapting the sending rate to the average long-term throughput of TCP, model-based congestion control can produce much smoother rate changes that are better suited for the aforementioned type of traffic.

Binomial Congestion Control [8], is a generalization of AIMD control laws. There are two parameters k and l in its control equations. So AIMD is simply one point in its parameter space.

Receiver-driven congestion control is used together with layered congestion control approaches. Here, the receivers decide whether to subscribe or to unsubscribe from additional layers, based on the congestion situation of the network.

All of the schemes cannot scale well with respect to network throughput and delay. Oscillation is a big problem as shown in Figure 2.8 (I) for the window size changing of the TCP mechanism. Especially, AIMD scheme results in a rate that displays the typical short term saw tooth like behavior shown in Figure 2.8 (II).

Furthermore, the end-to-end congestion control has the disadvantage that it relies on the collaboration of the end systems. Experience in the current Internet has shown that this cannot always be assumed: greedy users or applications may use non-TCP-friendly

mechanisms to gain more bandwidth.

2.4.4 Dynamic Buffer Control

Dynamic buffer control employs network elements to participate in both congestion detection and congestion recovery. Its target is to maintain the buffer size at some preset level. It is well known that dealing with congestion after it is first detected, is more effective than letting it gum up the works and then trying to deal with it. This observation leads to the idea of discarding packets before all the buffer space is really exhausted. A popular algorithm for doing this in IP data network is called *Random Early Detection* (RED) [45]. By letting routers drop packets before the situation has become hopeless, the idea is that there is time for action to be taken before it is too late. This strategy is worked only when the source knows that the lost packets are generally caused by congestion and it will respond by slowing down instead of trying harder.

Braden [19] suggests Active Queue Management (AQM) as a mechanism for congestion control and looks RED as one of the possible algorithms. After RED was first proposed in 1993, many AQM based approaches such as Adaptive RED (ARED) [41], Stabilized RED (SRED) [84], BLUE [42], and Adaptive Virtual Queue (AVQ) [71] have been proposed and their performance has been evaluated by Firoiu [43].

In these approaches, the TCP/AQM flow dynamics are modeled and analyzed in terms of feedback control theory. In order to control bursty Internet traffic effectively and efficiently, most proposals recommend using the average queue length as a congestion indicator. AQM algorithms are designed to decrease the response time and improve the stability and robustness of the control loops by regulating the queue length to agree with a desired value. It can be modeled as shown in Figure 2.9.

In this model, the feedback control system consists of:

1. A plant, which represents a combination of subsystems such as TCP sources, routers, and TCP receivers that send, process, and receive TCP packets respectively.

2. An AQM controller, which controls the packet arrival rate to the router queue by generating a packet drop probability as a control signal.

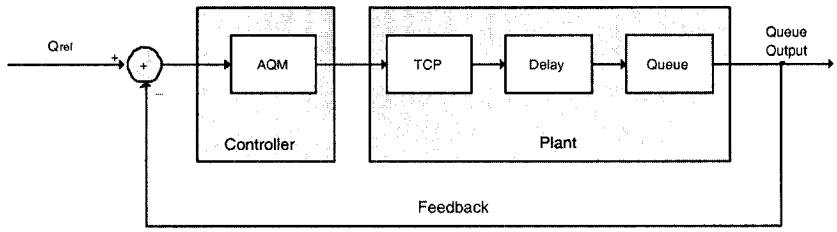


Figure 2.9: Control Theoretical Approach for AQM

3. The queue length at a router as a plant variable (i.e., a controlled variable) denoted by Q_t.

4. A desired queue length at a router (i.e., the reference input) denoted by Q_{ref}.

5. A feedback signal, which is a sampled system output (i.e., queue length) used to obtain the error term, $e(t) = Q_{ref} - Q_t$.

Firoiu and Borden [43] model the RED algorithm as a feedback control system and derive fundamental control laws governing the traffic dynamics in TCP/IP networks. Then, they recommend rules for the configuration of the RED control law such as a drop-conservative policy and a delay-conservative policy. In addition, they derive a set of recommendations for configuration of RED queue length averaging mechanisms such as the sampling frequency and the averaging weight.

A liberalized TCP/AQM dynamic model is developed and analyzed especially for TCP/RED dynamics in terms of feedback control by Hollot [61]. The same author introduces Proportionl-Integrational (PI) controller into the control system [60], which has been proposed both to improve responsiveness of the TCP/AQM dynamics using the P part and to stabilize the router queue length around Q_{ref} using the I part.

However, RED shows oscillatory behavior and gives poor performance under a wide range of traffic environments. Furthermore, RED shows sluggish response to the traffic dynamics. Christiansen [27] concludes after an extensive study of RED that it does not provide any benefits over tail-queuing disciplines.

Instead of dropping packet, Floyd proposes Explicit Congestion Notification (ECN) [44], in which routers signal congestion by marking ECN packets. Receiver echoes the mark back to sender in the next ACK. The Sender clears the mark after it makes

adjustment. Hollet [61] uses one bit marking strategies for Internet congestion control. At best, the one bit marking algorithms can match bandwidth to capacity only in the mean and requiring large buffers.

The previous schemes mainly focus on the TCP/IP network. Some other studies model and analyze a general control system on single link for packet switched network. With control theory, algorithms are designed to increase the speed of response and improve the stability and robustness. Lotfi Benmohamed [10] develops an analytical method for the design of controllers that ensure good dynamic characteristics along with some fairness in resource allocations. In his paper, the level of network congestion is monitored through the occupancy x of the buffer with the control target being some threshold value x_0. Based on the difference between x and x_0, the congestion controller associated with each link periodically calculates an admission rate and supplies rate to the lowest level allowed by intermediate links. Adaptive and robust controller are designed and verified. He proposes and studies Proportional (P) and Proportional-Derivative (PD) controllers applied to the switch's queue size. It makes a significant early contribution in plant modeling. Blanchini [16] presents the robust PID control theoretic framework that observes the buffer length and tries to maintain it at a preset level through rate control. Two important issues must be addressed by any feedback-based scheme: the fairness criteria and the time for the algorithm to converge. At the onset of congestion, reactive control instructs the source nodes to throttle their traffic flow by giving feedback to them. A major problem with reactive control in high-speed networks is slow feedback. As shown, the effects of high speed channels make the overhead significantly due to the propagation delay, therefore by the time that feedback reaches the source nodes and the control is triggered, it may be too late to react effectively. There is a possible improvement technique to overcome the difficulty caused by slow feedback. If reactive control is performed between network users and the edge of the network [49], the effect of propagation delay may not be significant since the distance feedback information propagates is short. However, this limits the reactive control to the edge of the network.

All of the prior studies from a control theoretic perspective use the buffer length as their congestion indication. Therefore, the resulting plant has a fast dynamics and as

such, the applicability of such schemes is doubtful, due to the delays incurred. Further, the buffer size is not a QoS parameter and it cannot be used in the SLA to provide differentiated services. On the other hand, the buffer size is usually preset by the equipment vendors and the vendors do not provide access to dynamically control the buffer length. Moreover, from practical reasons, modifying dynamically the buffer length is impossible. First, equipment vendors do not allow this. Secondly any reconfiguration of any network equipment ends in large amounts of traffic loss during the time periods of reconfiguration in most of cases.

2.4.5 Packet Loss Control

Since in most statistical QoS environments, the packet loss is the premier QoS parameter, the packet loss probability is most appropriate control signal, which can be feed backed and controlled to satisfy with a preset target. This signal has a slower dynamics compared with the buffer length.

Lambiri [72] gives the theoretic research and some simulation results in his thesis for the packet loss control issue. The basic idea is that the packet loss probability in the core network can be maintained to be below the preset value through throttling the ingress traffic in the Provider Edge (PE) node. His dissertation presents a service aware network control platform, which focuses on the *quantitative* QoS aspects of the VPN services, as defined by their SLA and topological characteristics, rather than the *qualitative* service characteristics. In the platform, the *proactive* and *reactive* control strategies are used in complementary ways, with the ultimate goal of maximizing the network utilization subject to the QoS constraints.

Firstly, using the theoretical framework of large deviations, he proposes approximations for packet loss probability of the aggregate traffic at FIFO and PQ schedulers with finite buffers. Then applying the loss probability equations, he derives three loss estimators for FIFO and PQ schedulers, where only the aggregate traffic that arrives at each queue has to be known.

Secondly, the concept of service admission control (SAC) is developed. This concept represents a generalization of the connection admission control at service level, which arises from the need of atomic admission or rejection of a service at a node in the network

during the service setup phase. QoS-VPNs allow the aggregation of several QoS classes under one service, and as such, the service admission has to span several of these classes. Hence, the admission control mechanism will accept or deny the addition of a *service* at the node, rather than a simple connection. It presents an accurate algorithm for SAC based on model correction through packet loss probability observation. The observation of packet loss is based on the assumption that the link capacities are large, and therefore in a sample interval that is greater than 1 second loss can be observed fair degree of accuracy.

Lastly, it presents a network control system that dynamically actuates the VPN ingress traffic in order to maintain the loss experienced at the core nodes below a given target. Reactive control strategies are used to control the network congestion that appears because of unforeseen traffic behavior.

Unfortunately, his research result is only based on simulations. The actual performance is not verified on the live network. Moreover, for the feedback control scheme, he just proposes a simple idea and does not give detailed analysis.

2.5 Conclusion

With the analysis of prior studies in the literature, the existing QoS and congestion control mechanisms have the following drawbacks:

- Direct admission control needs accurate description of traffic and that is inapplicable in the operational networks. Furthermore, in order to guarantee QoS it wastes the sources and causes a low utilization of network resources through the reservation.

- Measurement based admission control can increase the network utilization based on measurement and not need clients to provide accurate information about the traffic, but it cannot avoid congestions and QoS violations because of the unpredictable traffic.

- Implicit feedback control only dependents on end system, thus it has a huge delay and low efficiency in controlling congestions. Besides, the AIMD mechanism will cause oscillating problem.

- Buffer control mechanism tries to maintain the queue length, but the resulting plant has fast dynamics. Due to the delay incurred, its applicability is doubtful.
- ABR service tries to adjust the input rate to match the available bandwidth through explicit rate control. However, it only tries to provide the bandwidth assurance.

Lambiri gives the theoretic research and some simulation results in his thesis for the packet loss control issue, which will be the basis for this further study. Our thesis will extend his work in the following direction: First, based on large deviation theory and the traffic characteristics of VPN services, new packet loss probability estimators are proposed and their performance is evaluated in a live network with live measured traffic. After analyzing estimation errors, a Reactive Estimator (RE) is constructed to adapt itself to the variable contexts, where a Kalman filter is employed to estimate the mean and variance of the packet rates. The SAC algorithm is evaluated in the live network, which proves the feedback control scheme is the focus of the research and studied in details. In particular, the analytic model is inappropriate for the control due to its simplicity. Therefore, the dynamic packet loss system is identified by applying the RPE system identification technology. The input-output and state based controllers are designed for the identified nominal model and the system stability is analyzed. However, the identified model has uncertain parameters. The robust control technology is applied to satisfy the requirements.

Chapter 3

QoS Control Framework for the VPN Services

This chapter presents the QoS control framework for the VPN services. The framework is based on an out-of-box architecture that assembles proactive and reactive control schemes into one integrated system. The purpose of this chapter is to give an overview of the proposed QoS control architecture.

Service Admission Control (SAC) process is employed to proactively protect the network from congestion. In particular, when a new VPN service request is coming, it is accepted only if it can pass the service admission check, where the estimated QoS value is compared with the preset QoS parameter criterion. By limiting the number of VPN services that the network can afford, the QoS for the provisioned services is assured. However, the VPN services usually have a longer life span than the original call connections. Once a VPN service is accepted, the traffic is still subject to change and be unpredictable. Therefore, SAC is not enough to completely control the QoS.

In such a case, the *dynamic control system* is utilized to maintain the packet loss probability targets for all of the instantiated services. In order to achieve the goal, the control process dynamically adjusts the ingress rates for selected VPN services on the PE nodes when traffic congestion occurs and the loss target is not reached in the core network. Through this, congestion may be controlled and the loss target may be maintained for the VPN services.

In the following sections, the overall picture of the thesis and MPLS VPN services

are first reviewed. Then, the Qos control architecture is introduced, where the emphasis is put on the dynamic control system. Finally, the system is implemented and one of the components, traffic monitor, is demonstrated.

3.1 High Level View of the Control of Packet Loss Probability for VPN Services

In this section, the main concepts which are at the foundation of all research presented in this thesis are introduced and explained. A high level description of the Control System for Packet Loss Probability and the alternatives for designing appropriate control loop components are discussed.

VPN is an important service offering for any network service providers. It can be justified that in a metro environment some networks will carry only such traffic. Metro networks are characterized by small link spans, which make the propagation delays negligible. In such environment, packet loss is the primer QoS parameter to be estimated and controlled. This is the reason why a packet loss control is proposed, studied, designed, implemented and tested practically in a real production network.

The basic premises of the work undertaken in this research presented in this thesis are that the controlling system for packet loss is focused on the core network administrated by network service providers. They will provide VPN services with well defined QoS behavior specified by the long term packet loss probability as per the SLA. Although the customers side QoS control is very important, it is not in the focus of this thesis.

The main idea of the control system for packet loss probability stems from the fact that the packet loss probability is a very low varying parameter when compared with the speed of the packet loss in core networks. This can be so fast that from a practical point of view does not present any *inertia* at all. It is obvious that there is no system in this world which can interact that fast with the controlled object, in this case the network, such that packets which can be lost in fractions of microseconds can be compensated for in real-time. However, it is obvious that the packet loss probability, which is the guaranteed parameter in any SLA, is a low varying variable as it is calculated over long periods of time. The other important fact in regards to choosing the packet loss

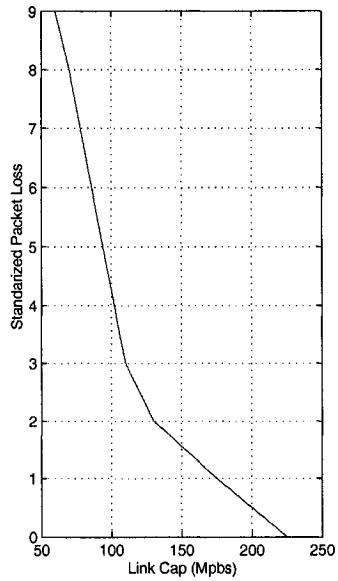


Figure 3.1: Packet loss probability vs. link utility

probability as the controlled parameter, is that based on the following diagram 3.1, measuring the packet loss probability for aggregated flows and taking actions on the admission rate of each user, can bring back the network in its guaranteed SLAs in a very short time as compared with a long time needed for the packet loss probability to vary.

Following the same rational, the packet loss probability can be compensated in real-time if the packet loss rate is spread on all the number of users contributing to the core aggregated flow. For example if every user is given 100 Mbps bandwidth by reducing, for example the traffic with 1% in an aggregated flow of 1 Gbps, the core has 10 Mbps more used to reduce the packet loss. Thus the control strategy proposed for maintaining the packet loss probability in the core network is to act on the ingress traffic rates from different VPN customers which are adjusted on the provider edge nodes.

To design a complete control system for the packet loss probability it therefore necessary to first build a model for the controlled process, to design appropriate estimators which will be the feedback transducers of the control system, to proceed to the identification of system model parameters, to design controllers and build actuators to control and guarantee the packet loss probability in the core network.

Building a control system for packet loss probability control requires therefore, to

investigate and choose a guiding mathematical model for the packet loss in core links as a controlled parameter. The literature of control approaches to communications all use the dynamic buffer model. This model has also been chosen in this thesis and it will be expanded in Section 6.

As the packet loss probability is the output parameter of the system, there is a need for a transducer to measure it in real-time. The thesis reports on the efforts made to find the best transducer possible.

Packet loss probability is firstly estimated based on the theory of large deviations, where the aggregated core traffic is assumed as a Gaussian type. However, in the practical network, the buffer size is limited and small. Moreover, the Gaussian traffic model sometime may not describe the traffic exactly. Therefore, the estimation errors are unavoidable. After analyzing the errors, technologies such as the reactive estimation and Kalman filtering are developed to overcome these errors obtained via measurements. Specifically, the reactive estimator and one-step forward Kalman filter are combined to adjust the estimator behaviors according to the online measured data. With such an approach, the packet loss probability is estimated more accurately.

The packet loss probability is slower than the dynamic buffer occupancy and can be controlled by the closed loop system. However, the analytic model is too complex to be applicable in the practice. The packet loss probability model is identified by applying the identification theory on the real measured data. Again, the model has some unavoidable uncertainties, either because of the linearization process or the inaccurate traffic model assumption. To solve these uncertainties, the recursive prediction error method is also studied and applied to identifying the control system parameters.

Based on the identified system and on a good transducer technique, a controller for packet loss probability can be devised. A series of classical input-output and input-state-output controller synthesis techniques were applied to study the quality and performance of the controls of packet loss probability parameter in the core networks are studied. From studying the variation of identified parameters, it was concluded that even though with RPEM, the final packet loss probability system still has a certain range of uncertainties in the way the control system parameters are obtained. To solve such possible situations the control system science has introduced the robust control of system with

randomly variable parameters. In this thesis, the robust and optimal controller was designed to compensate for the uncertainties in the model.

In order to provide the reader a continuity in the logic of the control techniques and technology as applied to packet loss probability in core network links carrying flows generated by VPN based services, all of the above components together and give a whole picture of the modeling, estimation, identification and controlling strategies used.

3.2 MPLS VPN Services

In a general IP network, packets of the same connection usually do not traverse the same path, because of the connectionless IP protocol. It is difficult, therefore, to guarantee QoS for the provided VPN services. The emerging MPLS technology makes it possible to solve these problems, by which end-to-end VPN services can be provided.

3.2.1 Introduction to MPLS

Multiprotocol Label Switch, or MPLS, is rooted in many of the early tagging and label swapping protocols. To present an incomplete list, Ipsilon's IP Switching, IBM's Aggregate Route-based IP Switching and Cisco's Tag switching. All of them developed proprietary approaches to forward data by using labels.

Among the many positive attributes MPLS brings to networking is the ability to provide connection-oriented services to the inherently connectionless IP networks. An end-to-end MPLS connection is called a Label Switch Path (LSP). This connection may be established for a variety of purposes, such as to guarantee a certain level of performance, to route around network congestion, or to create IP tunnels for network-based virtual private networks. In many aspects, LSPs are no different from switched paths in ATM or Frame Relay networks, except that they are not dependent on a particular Layer 2 technology. Traffic is assigned to LSPs based on pre-defined criteria such as destination IP address, TCP/UDP port number, VLAN Identifier, DiffServ, or 802.1p priority. For example, all high priority traffic destined for a critical application server may traverse an LSP dedicated to this traffic.

Information about the connection is summarized into an MPLS label, which is in-

serted between the Layer 2 and Layer 3 headers of each packet. Labels enable different paths to be established for different customers, or even for different applications for a single customer. Labels are assigned and distributed via a protocol used for this purpose.

The first label is added to an incoming packet by a Label Edge Router (LER), also called an Edge LSR. Switching with labels is a simple indexing mechanism that replaces traditional Layer 2 (Ethernet/ATM) or Layer 3 (IP) packet forwarding with fast, simple switching. At each hop in the network, a router examines the incoming label to figure out the next forwarding hop for the packet. This scheme eliminates intensive address lookups that reduce overall packet throughput and limit scalability.

Along the path, each Label Switch Router (LSR) makes forwarding decisions based solely on the contents of the label. At each hop, the LSR strips off the existing label and then it applies a new label which tells the next hop LSR how to forward the packet. All MPLS routers within the network regularly exchange label and reachability information to build a complete picture of the network. The picture of the network is then used to determine paths and specify the new label to place onto packets.

3.2.2 MPLS VPN Services

MPLS VPN Service, in particular BGP/MPLS VPN, is a beneficial application based on MPLS technology. BGP/MPLS VPNs provide substantial value to the customer in the form of increased simplicity, great flexibility, and outsourced routing, while representing a significant revenue stream for the NSPs.

Figure 3.2 shows one typical MPLS VPN network. To deliver the MPLS VPN services, the data communication network under consideration consists of a number of the following nodes:

- Customer Edge (CE) routers, which are associated with customer sites and usually managed by the customer.
- Provider Edge (PE) routers, which serve as the customer's entry and exit points for the VPN and are managed by the provider. Most of the VPN functionality of the BGP/MPLS solution is concentrated in the PE routers.
- Provider (P) routers, which form the core of the provider network and are primarily

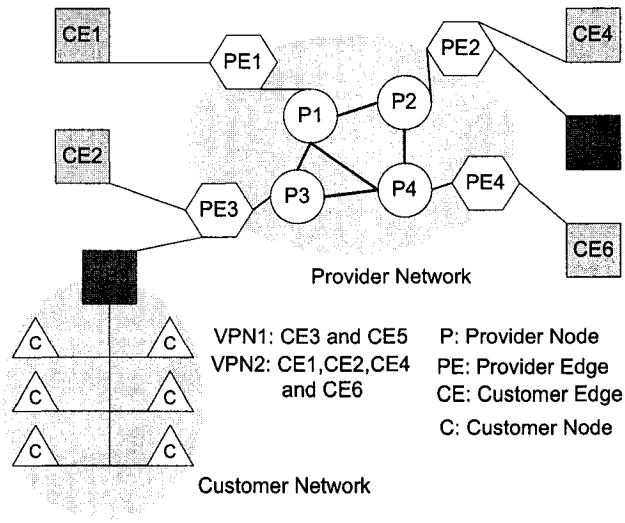


Figure 3.2: Topology Model for MPLS VPN Service Based Network

concerned with forwarding VPN traffic that has been placed in MPLS frames by CE and PE routers. P routers are managed by the provider.

- Custom Node (C) routers, which are private nodes in the customer network.

When the VPN service is set, an LSP between the PE nodes is automatically established on the IP topology of the P node network. The setup phase can be done through a protocol that distributes the labels such as the Label Distribution Protocol (LDP). The label is a short, fixed-length, locally-significant identifier which is used to identify a Forwarding Equivalence Class (FEC), while FEC is a group of IP packets that are forwarded in the same manner, over the same path and with the same forwarding treatment. After the label distribution phase is completed, transmission of the data using label switching within the established LSP starts. Consequently, the VPN service in the provider's network can be managed in secure and connection-oriented mode.

In the above Figure 3.2, the service provider has provisioned VPN services to two customers: VPN1 and VPN2. VPN1 includes the private connection between CE3 and CE5. To build up the VPN1 service, the service provider has created one LSP in the MPLS VPN backbone as follows:

$$CE3 \Longleftrightarrow PE3 \Longleftrightarrow P3 \Longleftrightarrow P1 \Longleftrightarrow P2 \Longleftrightarrow PE2 \Longleftrightarrow CE5$$

where $CEi \Longleftrightarrow PEj$ means the label switch path between two routers. This LSP between CE3 and CE5 is one of many possbile configurations.

In the same reason, there are 6 LSPs created to make the point-to-point VPN concetions between CE1, CE2, CE4 and CE6 for VPN2.

$$CE1 \Longleftrightarrow PE1 \Longleftrightarrow P1 \Longleftrightarrow P2 \Longleftrightarrow PE2 \Longleftrightarrow CE4$$

$$CE1 \Longleftrightarrow PE1 \Longleftrightarrow P1 \Longleftrightarrow P4 \Longleftrightarrow PE4 \Longleftrightarrow CE6$$

$$CE1 \Longleftrightarrow PE1 \Longleftrightarrow P1 \Longleftrightarrow P3 \Longleftrightarrow PE3 \Longleftrightarrow CE2$$

$$CE2 \Longleftrightarrow PE3 \Longleftrightarrow P3 \Longleftrightarrow P1 \Longleftrightarrow P2 \Longleftrightarrow PE2 \Longleftrightarrow CE4$$

$$CE2 \Longleftrightarrow PE3 \Longleftrightarrow P3 \Longleftrightarrow P4 \Longleftrightarrow PE4 \Longleftrightarrow CE6$$

$$CE4 \Longleftrightarrow PE2 \Longleftrightarrow P2 \Longleftrightarrow P4 \Longleftrightarrow PE4 \Longleftrightarrow CE6$$

where $CEi \Longleftrightarrow PEj$ means the label switch path between two routers.

3.2.3 MPLS VPN QoS

There are two fundamental models that are used to define MPLS VPN QoS. Each offers diverse service flexibility to both the user and the provider. The first one is called the *pipe service model* [62], where traffic is specified between pairs of nodes. The second is *hose service model* [37]. In the hose model, only the total ingress and egress traffic from each node to the provider network is specified. With the *pipe* model it is possible to make QoS guarantees for *all* the traffic that belongs to a service instance; this is not the case for the *hose* model. Hence, the improvement in bandwidth utilization [37] from deploying the *hose* model versus the *pipe* model is at the sacrifice of a lesser guarantee semantic. In order to provide the quantitative QoS to the user, only the pipe model is considered in the thesis.

In this section, packet loss differentiated VPN services are proposed in the service provisioning model. The MPLS VPN network will support a number of Classes of Service (COS), associated with the statistical Quality of Service and denoted by $\mathcal{C}$. Individual COS ($\mathcal{C}_j$), is defined by a loss guarantee parameter P_j. P_j is the packet loss probability and is viewed as the upper bound of loss experienced by the traffic of class

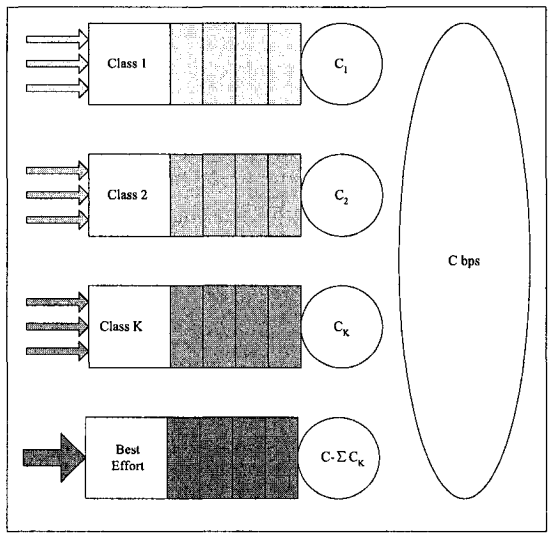


Figure 3.3: Loss Differentiated Service Model

$\mathcal{C}_j$ in any part of the network over a period of time T_j. P_j is satisfied by the following equation:

$$0 \leq P_j \leq 1 \tag{3.1}$$

Consider the output port of a P node, where the service rate is c bps. The port is assumed to support K QoS traffic classes and one Best Effort (BE) traffic class, each having its own logical queue, as shown in Figure 3.3. For each type of service $\mathcal{C}_j$, the associated queue scheduling has a minimum service rate c_j, where c_j is satisfied with the following equation:

$$c = \sum_{j=1}^{K} c_j \tag{3.2}$$

The model describes the DiffServ/MPLS network framework, in which QoS is differentiated on a per-class basis. Hence, each class of service $\mathcal{C}_j$ is completely defined by the two QoS parameters $< c_j,\ P_j >$ in the provider network; made precise by the following equation:

$$\mathcal{C} = \{\mathcal{C}_j \mid \mathcal{C}_j =< c_j,\ P_j >,\ j \in [1, K]\} \tag{3.3}$$

Each COS $\mathcal{C}_j$ is assumed to be served separately by utilizing technologies, for example WFQ [11], thus the different classes are controlled independently. In order to make the concept and notation simple, only one class of service is considered.

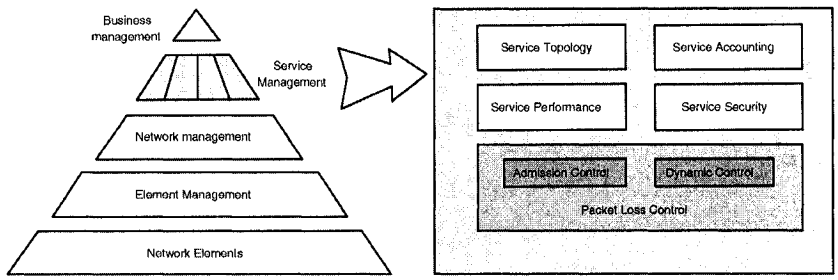


Figure 3.4: TMN Model and the Relationship between Network Management, Service Management and Business Management

The thesis will focus on the case of a single queue, where FIFO scheduling is used. For each node, the requirement is simplified to meet the preset packet loss probability for all of the VPN services constrained by a service rate c. When $K = 1$, (3.3) is simplified as follows:

$$\mathcal{C} = \{< c,\ P >\} \tag{3.4}$$

3.3 Architecture of the QoS Control System

The Telecommunications Management Network (TMN) standard [78] has identified that the functionality of managing a telecommunication network can be divided into four functional layers as shown in the Figure 3.4 (I): Network Element Management, Network Management, Service Management and Business Management.

The sphere of interest of this thesis belongs to the service management system, which is shown in detail by Figure 3.4 (II). The general VPN service management system includes functional blocks such as topology, performance, accounting and security. The focus is on the QoS control and guarantee for the VPN services, which is specified by the packet loss control. Two control mechanisms are combined together to address the packet loss control issues: Service Admission Control and Dynamic Loss Control. The control system is implemented in the following section.

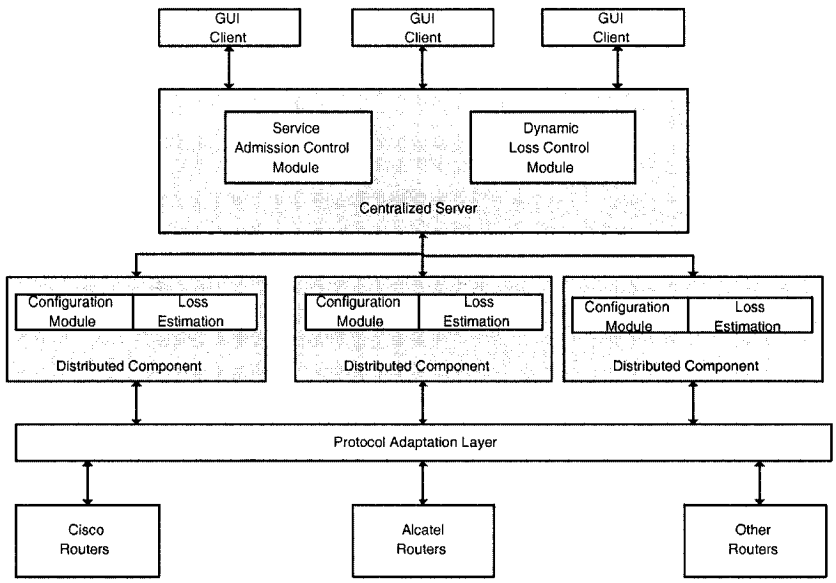


Figure 3.5: Block Diagram of the Loss Control System

3.3.1 Hybrid Centralized and Distributed Structure

A hybrid centralized and distributed approach to the packet loss control is shown in Figure 3.5. A centralized server maintains a view of the entire network and it is composed of two controllers which are responsible for making the final decisions. Some distributed servers run for measuring the local traffic and making the decisions based on part of the information. The system consists of multiple modules and runs in a hybrid centralized and distributed way. The multiple modules include: Packet Loss Measurement and Estimation (LM), Service Admission Control (SAC), Dynamic Loss Control (DLC), Protocol Adaptation (PA), and Configuration (CS). A brief description of each functional block is given below.

- *Packet Loss Measurement and Estimation.* Instead of calculating loss ratio directly, a loss estimator based on traffic measurement and large deviation theory is employed to calculate the loss probability, which will be explained in the next chapter. The block receives the traffic measurement from the protocol adaptation layer and description of the new service from the clients. After its calculation, LM will send the estimated packet loss probability to the service admission server and loss control server, respectively. LM will send a stream of data to the control

server for every sampling request generated by the latter.

- *Configuration.* The block is responsible for configuring the NEs. One of the main tasks is to set up the new VPN service when it receives the configuration command from the SAC module. The other one task is to dynamically execute the rate limit for some specific selected VPN services, which are decided by the DLC module according to the control mechanism. The configuration module generates the general CLI or SNMP scripts for the configuration tasks. The command scripts are then executed on the PE through PA layer.

- *Protocol Adaptation Layer.* In this thesis, Cisco, Alcatel, Foundry, Nortel, Hyperchip and Juniper devices are employed. However, different NEs support the same feature but using different CLI or SNMP syntax. This function is motivated by providing a generic framework for NE communication; i.e., it has a higher-level interface that allows the configuration and measurement of a like property (such as the measurement of packet rates) with the same command in a transparent manner. For example, the measured traffic is analyzed and generated in a uniform format whatever sFlow, netFlow or SNMP is employed.

- *Service Admission Control.* Service admission control is a generalization of the connection admission control problem to multi-classed service requests. The estimated loss probability obtained from the LM server can be used for implementing admission control criterion, establishing whether or not the further request of a new service can be accepted. The service admission result will be sent to the configuration server to implement the service or refuse the service. The service admission control strategy will be detailed in Chapter 5.

- *Dynamic Loss Control.* The function block dynamically controls the loss target. It receives the real time loss probability from the LM model and calculates the rate limit for the instantiated VPN services. It then sends these commands to the NE through PA Layer.

3.3.2 Dynamic Control Approach

The proactive service setup approach includes the path selection and the service admission control. However, the measurement based admission control system can give results deviating far from the preset loss targets; moreover, the VPN service lives longer than the call service. Service admission control is not enough if used solitarily.

This thesis emphasizes maintaining QoS targets after the service has been provisioned. A feedback control theoretical perspective is taken when examining the problem of packet loss control.

In the network model, packets generated by a VPN source node are delivered to their destination through a sequence of bottleneck nodes in the provider network. To manage the network, all the P and PE nodes in the path are considered. To simplify the analysis, only one single bottleneck P node is considered in the experimental analysis, which accumulates some PE connections. It is also assumed that the control system, which is located in a separate server, can receive traffic information from the bottleneck node by sampling the packet loss measurement & Estimation transducer and can send Command Line Interface (CLI) commands to the PE node.

The block diagram of the dynamic control system from a control perspective is depicted in Figure 3.6. The system consists of four basic parts: Packet Loss Probability Measurement and Estimation, Model Identification, Controller Design and Rate Actuator. Since in most statistical QoS environments the packet loss probability is the premier QoS parameter, the packet loss probability is directly used as the feedback signal. The packet loss probability is a very slow varying parameter when compared with the speed of the packet loss in core networks. The evolution of the packet loss can be so fast that from a practical point of view does not present any *inertia* at all. It means that by the time the packet loss is detected to the time the packet loss control system can intervene to correct the situation, the action of the control system is obsolete as the burst due to which the packet loss is caused disappeared already, thus the control action does not make sens. It is obvious that there is no system in this world which can interact that fast as the controlled object, in this case the network, such that packets which can be lost in fractions of microseconds can be compensated for in real-time. However, it is obvious that the packet loss probability, which is the guaranteed parameter in any SLA,

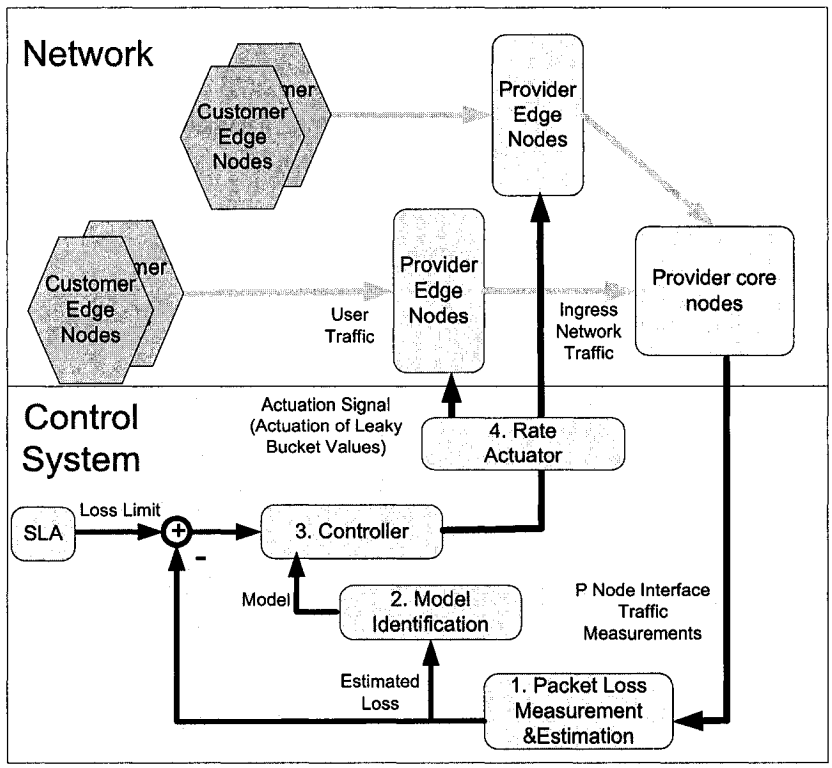


Figure 3.6: Block Diagram of the Dynamic Loss Control System

is a slow varying variable as it is calculated and it is accounted for over long periods of time. It is therefore, physically possible that through throttling the ingress traffic rate in the PE node, the packet loss probability in the core network is maintained below the preset value. The procedure to control the packet loss probability is thus as follows:

1. Packet Loss Measurement and Estimation. To dynamically control congestion and maintain the packet loss in the provider's network, the traffic in the P node is firstly monitored. Then, the measured traffic statistics (average and variance) are sent to the packet loss estimation process, which estimates the loss probability according to the equations in Chapter 4.

2. Model Identification. Based on the traffic measurement and packet loss estimator, the packet loss model is identified online.

3. Controller. The digital controller is designed based on the identified model. The estimated loss probability and preset loss target are then sent to the controller,

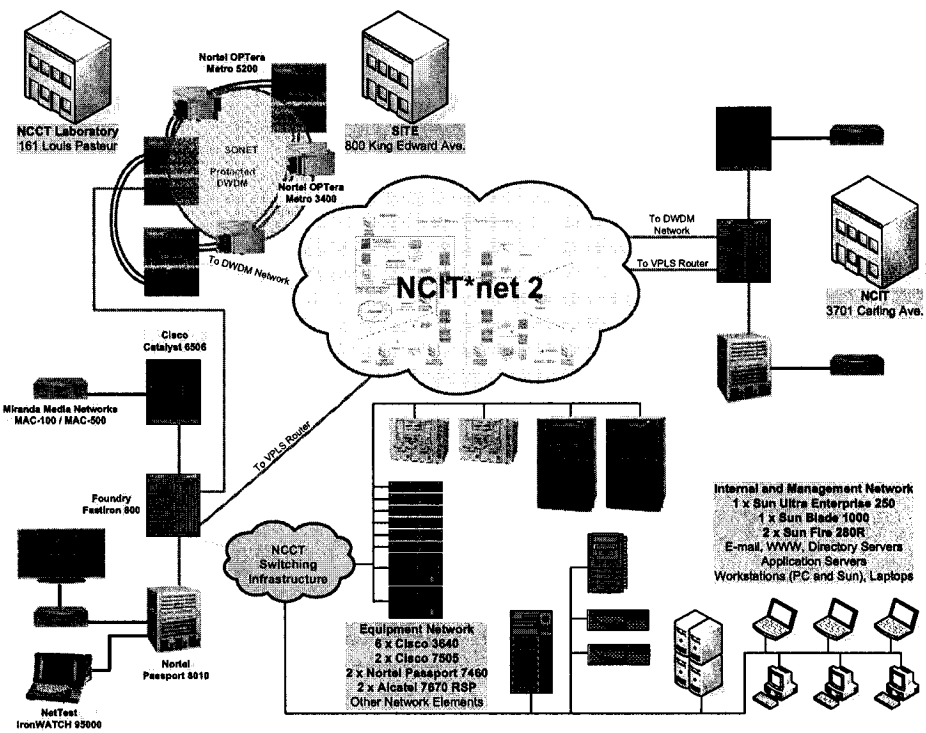


Figure 3.7: NCIT*net2 Network

where the maximum rate for each VPN service is calculated.

4. Rate Actuator: these explicit rate signals are adjusted by the rate limit actuator process and then sent to the PE node through CLI commands. These commands are executed and take effect on the PE node to throttle the VPN input rates. It is assumed that there is the traffic policer or shaper on the PE node.

3.4 System Implementation

The QoS control framework is implemented with a distributed computing architecture using Java language. The performance is evaluated on the live National Capital Institute of Telecommunications Network (NCIT*net2).

3.4.1 Introduction to the Live Network

All of the experiments in the thesis are designed in NCIT*net2 [39], while measurements are collected from the Network Computing and Control Technologies Research Laboratory (NCCT). The NCIT*net2 can be considered as a live networking laboratory to support research in networking at the physical, networking and application layers. The NCCT lab is equipped with the latest technologies used in internetworking, such as DWDM, SONET, ATM, Frame Relay, Ethernet, MPLS and IP. The topology of the whole network is shown in Figure 3.7.

The University of Ottawa provided NCCT with the special status of a research network and allowed NCCT to build its own Autonomous Systems (AS). On the other hand, NCIT gave NCCT the leading role in designing and building the new architecture of the NCIT*net. Accordingly, the NCCT research team designed and implemented a fully functional next generation network which plays a double role of live research and live production network promoting the architecture, principles, protocols and services of Next Generation Network (NGN). A large network of Sun and IBM servers and workstations help to build network and service control software to verify the research hypothesis and test the industrial viability of scientific works. Network and technology simulators, packet generators and network analyzers are used in this thesis to obtain real-data from this real network. State-of-the-art video-on-demand and audio-video conference equipment allows for rapid visualization and demonstration of the thesis research goals in the practical network.

3.4.2 Software Implementation

The framework can be implemented with a distributed computing architecture using Java language and J2EE technologies [83, 85]. Its advantages are:

- *Interoperability :* The system is based on Java language and allows for the interworking of the different operating environments such as Windows, Linux and Unix.
- *Scalability:* Due its highly distributed nature, the system can be deployed in any arbitrary Internet networks.

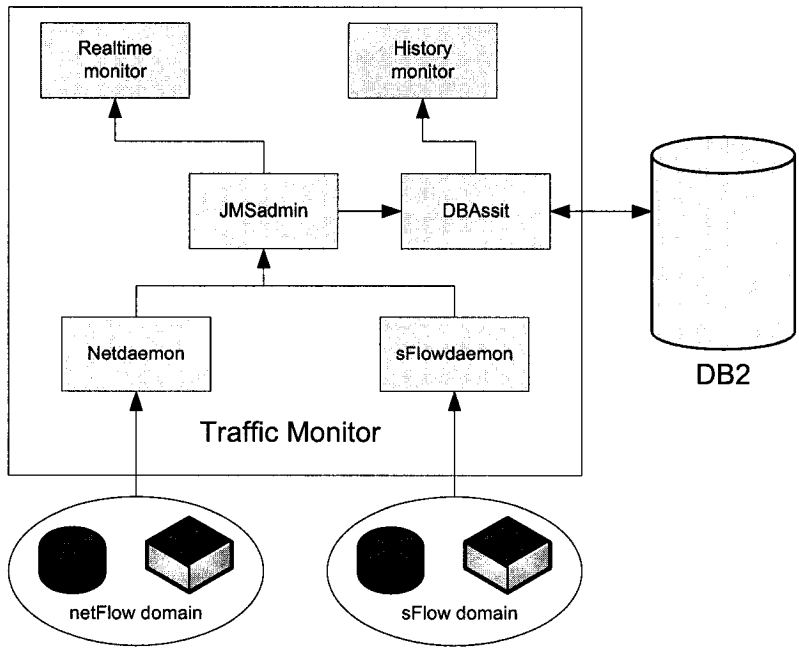


Figure 3.8: Block Diagram of Flow Traffic Monitoring System

- *Flexibility:* With the proxy structure, the system can support multiple protocols. It is very flexible to add support for a new protocol.

A subset of the framework, traffic measurement and monitoring, is implemented as a proof-of-concept. The system is developed on Windows NT 2000 operating system with IBM WebSphere 5.0, IBM DB2, JMS and Java language. sFlow technology is configured on the Foundry switch interface to generate traffic flow statistics. Netflow is configured on the Cisco routers. SNMP MIB II is polled for measuring the lost packets.

The diagram of the traffic flow monitor system is shown in Figure 3.8. The system is designed to transparently communicate with sFlow and NetFlow based flow meters. It is a distributed application which provides for interaction with multiple agents distributed in a large network. To implement the system, six processes are created: *Netdaemon* process, *sFlowdaemon* process, *JMSadmin* process, *Realtimemonitor*, *Historymonitor* and *DBAssist*. All of the six processes can run on different machines. *JMSadmin* process must first run before the other processes can run. It is responsible for managing and registering JMS service. JORAM is an open source software

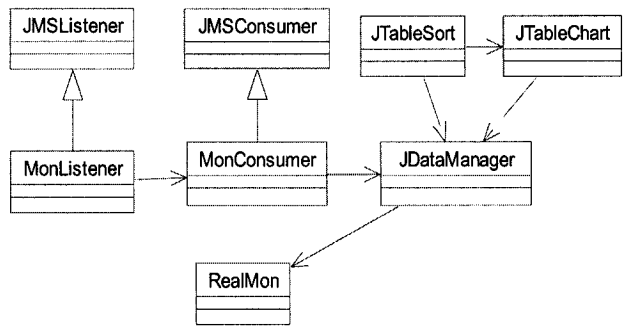


Figure 3.9: High-Level Class Diagram for the Real-time Monitor

used for deploying the JMS message. The other five processes can run independently and all of them communicate with Java Messages Service (JMS). *Netdaemon* process receives the NetFlow report, converting the data, and sending the common flow traffic information to *Realtimemonitor* and *DBAssist*. *SFlowdaemon* process receives the sFlow report, converting the data, and sending the common flow traffic information to *Realtimemonitor* and *DBAssist*. *Realtimemonitor* is to display the flow traffic information through GUI. The *Historymonitor* is responsible for displaying history flow information through retrieving data from the database. JFreeChart is used to generate the time series chart. *Historymonitor* displays history flow traffic information. It accesses the database with Hibernate technology. Because of the database, the traffic can be displayed for an extended time period, enabling the traffic to be analyzed in a different time scale. *DBAssist* is for the DB2 database access with Hibernate OODB technology.

The class diagram for the real-time monitor is shown in Figure 3.9, where classes are designed to display real-time flow traffic received from the network. Class *MonListerner* and *MonConsumer* are responsible for the communication with the *JMSAdmin* process. Class *MDataManager* explains the flow data and use class *JTableChart* and class *JTableSort* to show the the graphic traffic.

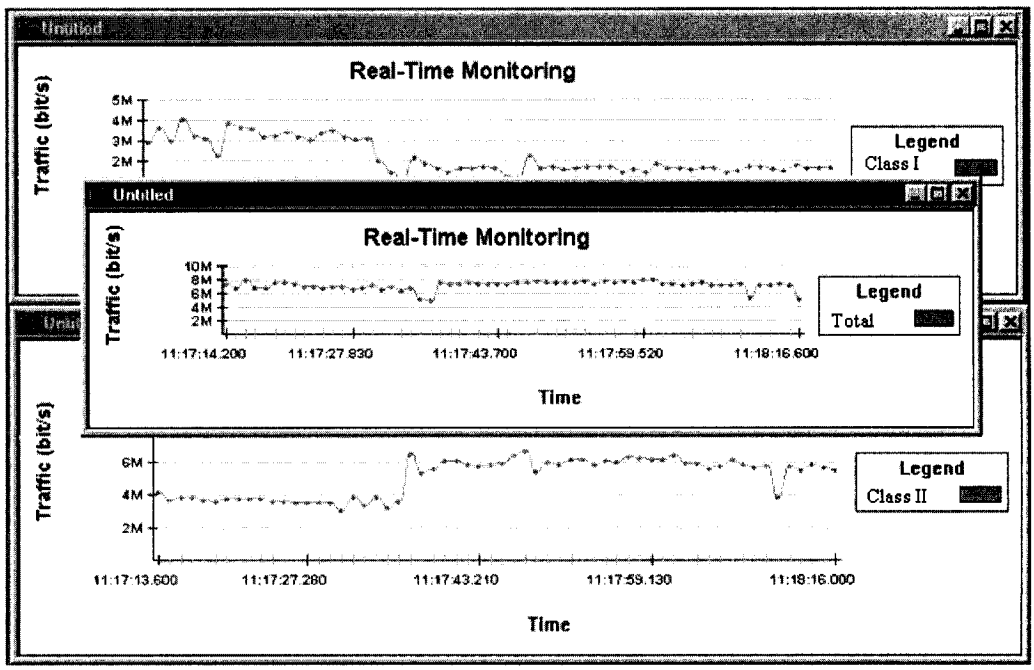


Figure 3.10: Result of Real-time Flow Traffic Measured On NCIT*net2 Network On Eth1/2 of Cisco1

The flow traffic monitor can display all of the MPLS VPN flows in the test bed network. One sample page is captured and displayed in Figure 3.10, showing traffic rates of each interval for the total-flow, each single flow and some aggregated flows.

3.5 Conclusion

This chapter presents the QoS control framework for VPN services in the provider's backbone network, where the service admission control and the dynamic control system are combined together. The implementation is proposed. As a proof-of-concept, the traffic measurement and monitoring system is implemented and demonstrated. The emphasis of the framework is the dynamic loss control system, which maintains the preset packet loss probability in the core network by avoiding congestion and providing long term statistical QoS for the customers. In the remainder of the thesis, each of the functional blocks in the system is designed and analyzed in a control theoretical perspective.

Chapter 4

Packet Loss Measurement and Estimation

When provisioning quantitative QoS for VPN services over packet switched networks, parameters such as packet loss, delay and delay jitter, besides the required bandwidth, have to be guaranteed. While the bandwidth is easier to guarantee, maintaining a value of the packet loss parameter below a preset value has serious difficulties, though is extremely beneficial. In the loss controlling system, one of the key issues is to accurately estimate the packet loss probability based on the characteristics of the input traffic. The issue has been largely studied, but there has been no simplification of estimators for practical applications. This chapter solves the packet loss probability estimation problem from the practical perspective and evaluates the solutions on the live NCIT*net2 network. In detail, the aggregated input traffic is modeled as a general Gaussian process, by assuming the VPN services multiplex a large number of flows. Two estimation formulae are derived by applying the Large Deviation Theory (LDT) on the buffer overflow probability. A series of experiments are designed on the live NCIT*net2 network to evaluate the performance of the estimators under different traffic arrivals and different buffer sizes.

4.1 Packet Loss Analysis

In this section, the packet loss is defined and analyzed by applying the large deviation theory. In order to make the concept and notation simple, only one class of service is considered and thus each network node implements the FIFO scheduling.

4.1.1 Packet Loss Definition

The input traffic fed into the FIFO scheduler can be viewed as a fluid flow, where packet boundaries are ignored. Let $A[0,t)$ be the amount of packets that arrives from all the sources in the interval $[0,t)$. Assume that $A[0,t)$ has stationary increments and let $\alpha(t)$ denote this stochastic increment process in a discrete time setting, where each interval has a time period T. The following equation is satisfied:

$$A[0,t) = \sum_{k=0}^{t/T} \alpha(k) \tag{4.1}$$

where each random variable $\alpha(k)$ is the number of packets that arrive during the time slot $[k,k+1)$[1] with a distribution $P_\alpha(n)$:

$$P_\alpha(n) = \mathbb{P}\{\alpha(k) = n\} \tag{4.2}$$

where $\mathbb{P}$ means the probability and n is a nonnegative integer.

By the similar way, let $q(t)$ be the stochastic process to denote the length of the queue. In a discrete time setting, let the random variable $q(k)$ denote the number of packets that occupy the buffer in the time interval $[k,k+1)$ with a distribution function $P_q(n)$:

$$P_q(n) = \mathbb{P}\{q(k) = n\} \tag{4.3}$$

The FIFO scheduler can be modeled as a state-space characterized dynamic system, where the state of the system is determined by the length of the queue. The state evolution can be described by (4.4).

$$q(k+1) = \min([q(k) + \alpha(k) - c]^+,\ b) \tag{4.4}$$

where $[x]^+ = max(0,x)$, b denotes the maximal finite buffer size and c is the link capacity for each time interval T.

[1]It is derived from $[kT,(k+1)T$, were T is the time for each slot and omitted for convenience

The amount of lost traffic in the interval $[k, k+1)$ is equal to the difference between the first term of the *min* expression in (4.4) and the second term of the same expression. Precisely, in the stochastic process of the packet loss, let the random variable $l(k)$ denote the lost packets in each interval $[k, k+1)$; then (4.6) holds.

$$l(k) = [[q(k) + \alpha(k) - c]^+ - b]^+ \tag{4.5}$$

$$= [q(k) + \alpha(k) - c - b]^+ \tag{4.6}$$

Since $l(k)$ is the instantaneous packet loss experienced by the multiplexer during each period $[k, k+1)$, it is not a very useful QoS metric. A more appropriate variable is defined in terms of the overall *packet loss probability* (P_{loss}), which is equal to the ratio between the expectation of the lost packets and the arrived packets for each interval. P_{loss} is expressed by the following formula:

$$P_{loss} = \frac{\mathbb{E}[l(k)]}{\mathbb{E}[\alpha(k)]} \tag{4.7}$$

where $\mathbb{E}$ is the mathematical expectation and $\alpha(k)$ is the random variable denoting the arried packets for each interval.

The packet loss probability in a finite buffer system is generally estimated by measuring the *packet loss rate* (plr), which is defined as the long-term ratio of the total measured lost packets to the total amount of traffic. *plr* is calculated as:

$$plr = \lim_{N\to\infty} \frac{\sum_{k=1}^{N} \hat{l}(k)}{\sum_{k=1}^{N} \hat{\alpha}(k)} \tag{4.8}$$

where $\hat{l}(k)$ is the measured packet loss in time interval $[k, k+1)$ and $\hat{\alpha}(k)$ is the measured input packets in the same time interval.

Although this approach is direct and simple, time averaging requires that an accurate estimation involves long computing time, especially for a low loss rate. The long-time delay is usually unacceptable in the real-time loss control system. Therefore, the estimation of packet loss probability is currently approached from another perspective, where P_{loss} is derived from the tail of the queue length distribution (tail probability) of an infinite buffer system. The tail probability, also termed as *buffer overflow probability* ($P_{overflow}$), means the probability that the queue size is larger than a fixed size b in an

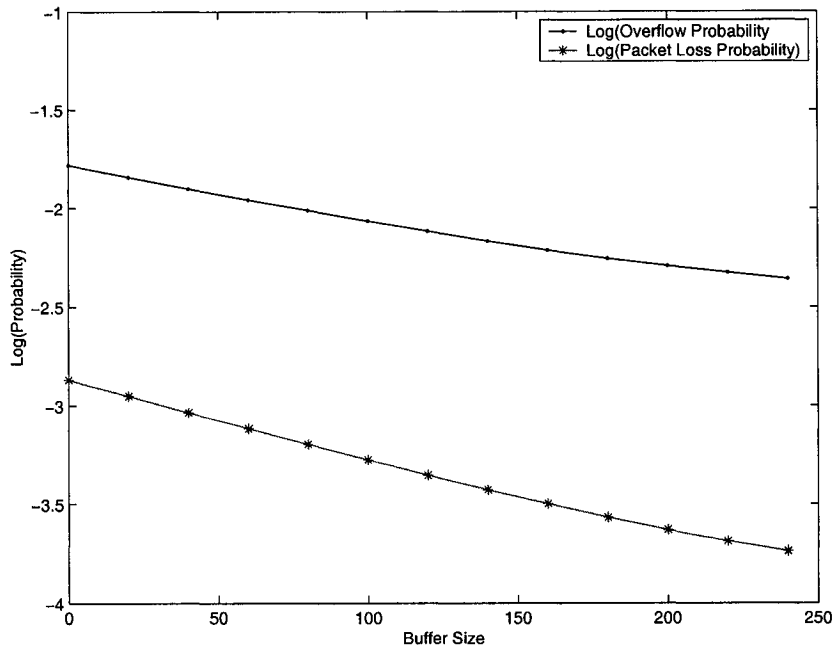


Figure 4.1: Comparison of Overflow Probability with Packet Loss Probability: Probability Vs. Buffer

infinite buffer system. It is expressed as:

$$P_{overflow} = \mathbb{P}(q(k) > b) \tag{4.9}$$

$$= \mathbb{E}[I(q(k) > b)] \tag{4.10}$$

where $I(x)$ is an identification function. In particular, $I(x) = 1$, if x is true; otherwise, $I(x) = 0$.

The buffer overflow probability are usually calculated as the amount of time the buffer size above level b divided by the total time. This can be expressed as:

$$P_{overflow} = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} I(q(k) > b) \tag{4.11}$$

Comparing (4.7) and (4.11), one knows that the overflow probability is averaged by time and the packet loss probability is averaged by the input. In general there is no relationship between these two quantities. However, the two probability-vs-buffer-size curves exhibit a similar shape shown in Figure 4.1 and the two probability-vs-utilization curves also show the same results in Figure 4.2, which motivate the study of estimating the loss probability from the overflow probability.

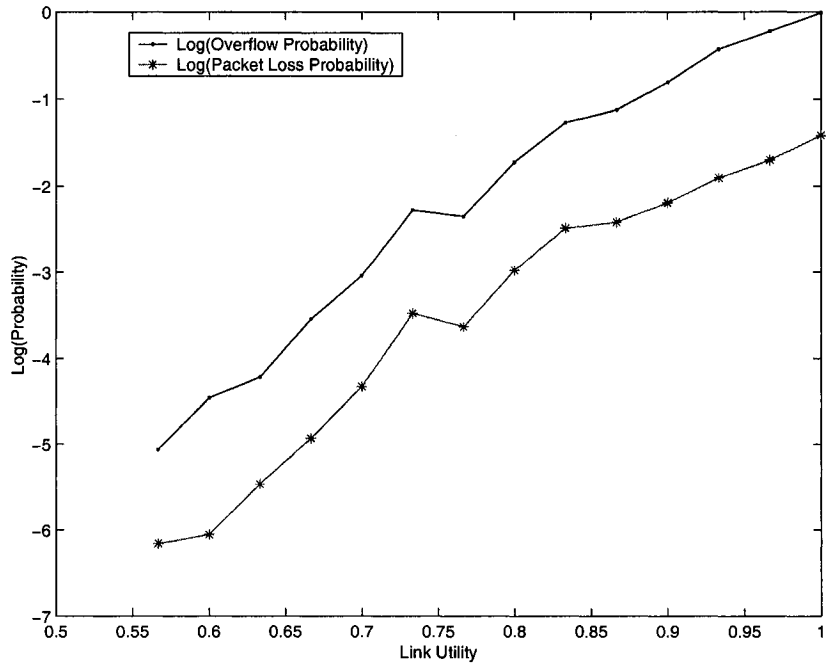


Figure 4.2: Comparison of Overflow Probability with Packet Loss Probability: Probability Vs. Utilization

In the following sections, two approximation techniques are analyzed, which will output a good estimate of the overflow probability. A simple integration calculation is then employed to compute the loss probability in order to make sure that the estimators are efficient in terms of calculation time but yet accurate enough for the packet loss control.

4.1.2 Large Buffer Based Analysis

For a general traffic arrival process, *Large Buffer Based Analysis* assumes that the buffer size is large (i.e., $b \rightarrow \infty$), and then asymptote is used to extrapolate the value of loss attained in the lower buffer region.

Let $A[0,t)$ be the amount of packets that arrives from the sources in the interval $[0,t)$. Assume that $A[0,t)$ has stationary increments. Kelly [67] defined the *effective*

bandwidth (EB), noted by $eb(\theta, t)$, of this stochastic process to be: [2]

$$eb(\theta, t) := \frac{1}{\theta t} ln\mathbb{E}[e^{\theta A(t)}] \tag{4.12}$$

where θ is the *space* parameter and t is the *time* parameter.

Parameter θ indicates the degree of statistical multiplexing: large values of θ indicate a low degree of statistical multiplexing and small values of θ indicate a large degree of statistical multiplexing.

Parameter t corresponds to the most probable duration of the buffer congestion period prior to overflow [67]. If time is increased asymptotically to infinity, the effective bandwidth becomes equivalent to:

$$eb(\theta, \infty) = \lim_{t \to \infty} \frac{1}{\theta t} \ln(\mathbb{E}[e^{\theta A(t)}]) \tag{4.13}$$

If Gartner-Ellis Theorem's [33] hypothesis is satisfied, then the existence of the limit of (4.13) is assured. By restricting the limit to a finite value c (i.e., the bandwidth), the overflow probability can be estimated by the following (4.14), which is proven by Chang [23].

$$\lim_{b \to \infty} ln\mathbb{P}(q(t) > b) = -\theta^* \tag{4.14}$$

where θ^* is the solution to the equation

$$eb(\theta, \infty) = c \tag{4.15}$$

The hypotheses of the theorem are not satisfied in all traffic cases. For example, in case of long-range dependent traffic, it has been observed that the queue process, if the buffer length is large, is not exponential. Results for this have been proven by Duffield and O'Connell [35] and Tsybakov and Georganas [109, 110]. However, it also has been shown by Ryu [100] and Heyman [59] that the small buffer behavior of long-range dependent traffic is close to an exponential queue decay. The result has important practical implications, because it means that the behavior of the long-range dependent traffic in small buffer settings can be still approximated by a logarithmically linear behavior.

[2] The semantic of the notation *ln* is natural logarithm and *log* is 10 based logarithm function.

Therefore, the tail probability or buffer overflow probability is exponential and can be described by:

$$\exists\ \zeta \in \mathbb{R}^+,\ \mathbb{P}(q(t) > b) = e^{-\theta^* b - \zeta} \tag{4.16}$$

In a discrete time environment, the similar conclusion also derived and expressed as:

$$\exists\ \zeta \in \mathbb{R}^+,\ \mathbb{P}(q(k) > b) = e^{-\theta^* b - \zeta} \tag{4.17}$$

Only the packet loss probability at the multiplexer is valuable, not the overflow probability. For approaching this, the loss probability at the FIFO multiplexer given by (4.7) can be calculated under the linear approximation. Through a simple integration calculation on the buffer size, the expectation of the lost packets in each interval ($\mathbb{E}[l(k)]$)is calculated as the following equation:

$$\mathbb{E}[l(k)] = \int_b^\infty \mathbb{P}(q(k) > x)dx \tag{4.18}$$

$$= \frac{e^{-\theta^* b - \zeta}}{\theta^*} \tag{4.19}$$

The packet loss probability is expressed with (4.19) divided by the expectation of the arrived packets for each interval. Therefore, the following (4.22) satisfies:

$$P_{loss} = \frac{\mathbb{E}[l(k)]}{\mathbb{E}[\alpha(k)]} \tag{4.20}$$

$$= \frac{e^{-\theta^* b - \zeta}}{\mathbb{E}[\alpha(k)]\theta^*} \tag{4.21}$$

$$= \frac{e^{-\theta^* b - \zeta}}{\mu\theta^*} \tag{4.22}$$

where θ^* is calculated as in (4.15) and μ is the mean of the arrived packets for each interval.

Applying the logarithm function to both sides of (4.22) gives the approximation of traffic loss probability under the large buffer assumption:

$$ln(P_{loss}) = -\theta^* b - ln(\mu\theta^*) - \zeta \tag{4.23}$$

The formula (4.23)will build the theoretical base for the large buffer estimator studied in the next section.

4.1.3 Aggregate Traffic Approximation

Another analytical approach is called Large Number of Sources Asymptotes (n-asymptotic), which is first proposed by Weiss [112]. The *n-asymptotic* assumes that the capacity of the link and the buffer length are scaled together in proportion with the number of sources that are fed to the multiplexer. Therefore, its result is based on homogeneity of the sources that feed the multiplexer. Obviously, this assumption is not applicable for the VPN services, since the sources of VPN are not identical. Moreover, the optimal algorithm of this estimation process is too complicated to be practical, which has been studied by Lambiri [72]. Therefore the *n-asymptotic* scheme is not investigated in this thesis.

Instead, *aggregate traffic approximation* is studied to reduce the computational complexity of the traffic estimation system, because in such an approach, the traffic is evaluated as an aggregate of the VPN sources. Only the loss for the aggregate is sought, so the calculation is simplified.

It is known from the Bahadur-Rao Theorem [7], that one can obtain the asymptotic tail distribution of the sum of n identically distributed random variables when $n \rightarrow \infty$. That is, if $a > 0$ and $X(k), 1 < k < n$, is a set of independent and identically distributed (i.i.d.) random variables, then the value of formula (4.24) exists.

$$\mathbb{P}(\sum_{k=1}^{n} X(k) > na) \tag{4.24}$$

In order to obtain the aggregate traffic approximation, identify na in (4.24) by (4.25):

$$na = (c + b) \tag{4.25}$$

where c is the maximum amount of traffic that the multiplexer can output over a period of length t, and b is the buffer size available at the multiplexer.

The formula (4.24) is rewritten and calculated as:

$$\mathbb{P}(A(t) > (c + b)) \approx \frac{e^{-I(c,b)}}{\theta\sqrt{2\pi\omega^2}} \tag{4.26}$$

and

$$I(c, b) = \theta^*(c + b) - log(\mathbb{E}[e^{\theta^* A(t)}]) \tag{4.27}$$

where θ^* is the point at which (4.28) is solved:

$$\frac{\frac{\partial \mathbb{E}[e^{\theta A(t)}]}{\partial \theta}}{\mathbb{E}[e^{\theta A(t)}]} = c + b \tag{4.28}$$

and ω is calculated by using (4.29):

$$\omega = \frac{\frac{\partial^2 \mathbb{E}[e^{\theta A(t)}]}{\partial \theta^2}}{\mathbb{E}[e^{\theta A(t)}]} - (c+b)^2 \tag{4.29}$$

Following the same line of reasoning with different initial hypothesis [74], the loss estimator can be derived. Assume that:

$$\exists! t_0 < \infty \quad s.t. \quad I_{t_0}(c,b) = \min_{t \in \mathbb{N}} I_t(c,b) > 0$$

and

$$\liminf_{t \to \infty} \frac{I_t(c,b)}{log(t)} > 0$$

then through the similar calculation as the large buffer based estimation in the previous section, the loss probability of the multiplexer is equal to:

$$P_{loss} = \frac{e^{-I_{t_0}(c,b)}}{\theta^2 \mu \omega \sqrt{2\pi}} \tag{4.30}$$

where ω is defined by (4.29) and θ is the solution to (4.28).

Applying the logarithm function to both sides of (4.30), it results the following:

$$ln(P_{loss}) = -\inf_{t>0} \sup_{\theta>0} I_t(c,b) - ln(\theta^2 \mu \omega \sqrt{2\pi}) \tag{4.31}$$

where inf is the minimization function and sup is the maximization function. The definition will be given as the followings:

- **Definition 4.1.** Let S be a set of real numbers. An upper bound for S is a number B such that $x \leq B$ for all $x \in S$. The supremum sup, if it exists, of S is the smallest upper bound for S. An upper bound which actually belongs to the set is called a maximum.

- **Definition 4.2.** Let S be a set of real numbers. A lower bound for S is a number B such that $B \leq x$ for all $x \in S$. The infinum of S denoted as inf, if it exists, is the largest lower bound for S. A lower bound which actually belongs to the set is called a minimum.

The main difficulty in evaluating (4.31) results from the constrained optimization problem of the first term of the right hand side. The *supinf* formula involves two optimization procedures: the first consists of, for fixed t, maximization with respect to θ, and the second a minimization with respect to t. The issues related to solving this problem will be discussed in next section.

4.2 Packet Loss Estimators for VPN Services

The previous estimators assumed that the traffic can be modeled as a stochastic process and the loss is derived from the moment generating function of the process. This section deals with the problem of simplifying the formulae if the VPN traffic can be modeled as a Gaussian model and the parameters can be identified through the traffic measurement.

4.2.1 VPN Traffic Model

To deliver the MPLS VPN services, the data communication network under consideration consists of a number of nodes introduced and shown in Figure 3.2 of the preceding chapter.

The VPN traffic is aggregated at each of the PE and P nodes and each of the VPN sources is itself an aggregate of the traffic generated by the VPN users, therefore hundreds or even thousands of VPN flows are more likely to be served in the core high-speed link with huge capacity. By appealing to Central Limit Theorem arguments, it is a safe assumption that the aggregate traffic in the core link is Gaussian traffic.

With a Gaussian Noise Model (GNM) [96], the traffic that arrives over an interval of length t is described by the equation:

$$A(t) = \mu t + Z(t) \tag{4.32}$$

where $Z(t)$ is normally distributed stochastic process with zero mean and μ is the traffic mean.

4.2.2 Single Loss VS Aggregate Loss

In the provider network, the FIFO scheduler aggregates a large number of VPN services. In order to improve the scalability, there is no single VPN flow info available in the core network and only aggregate traffic can be examined. Consequently, it is important to quantify the loss experienced by the individual VPN service and the loss of the traffic aggregate. To do this, consider the simplest case, where two sources are aggregated in the link: one for the individual VPN source that we want to study, whose arrival traffic is denoted by $A_1(t)$, and another that contains all the other VPN sources, whose traffic is denoted by $A_2(t)$. The total traffic $A(t)$ is then expressed as:

$$A(t) = A_1(t) + A_2(t) \tag{4.33}$$

Assume that $A_1(t)$, $A_2(t)$ and $A(t)$ are all Gaussian traffic, since each single VPN aggregates lots of individual traffic. Under these hypotheses, one can approximate the ratio of traffic loss by the first VPN source to the traffic loss in the second VPN sources, denoted by L_{ratio}, to be equal to the ratio between the traffic averages of the two sources. This can be expressed as:

$$L_{ratio} = \frac{\mathbb{E}[l_1(t)]}{\mathbb{E}[l_2(t)]} \tag{4.34}$$

$$\approx \frac{\mathbb{E}[A_1(t)]}{\mathbb{E}[A_2(t)]} \tag{4.35}$$

The loss probability experienced by the first VPN source, denoted by P^1_{loss}, should be equal to the loss probability of the total traffic, denoted by P^{total}_{loss}. This can be proved in the following:

$$P^1_{loss} = \frac{\mathbb{E}[l_1(t)]}{\mathbb{E}[A_1(t)]} \tag{4.36}$$

$$= \frac{\mathbb{E}[l_{total}(t)]}{\mathbb{E}[A_1(t)](1 + 1/L_{ratio})} \tag{4.37}$$

$$= \frac{\mathbb{E}[l_{total}(t)]}{\mathbb{E}[A(t)]} \tag{4.38}$$

$$= P^{total}_{loss} \tag{4.39}$$

Under the above assumptions, the loss probability of each VPN source is equal to the aggregate loss probability. Therefore, in the following section of the thesis, the packet loss probability will be investigated only for the aggregated traffic in the core link.

4.2.3 Large Buffer Based Estimator

The loss estimator based on the Gaussian model assumption will be derived in this section. For Gaussian traffic model in (4.32), the effective bandwidth is given by:

$$eb(\theta, t) = \mu + \frac{\theta}{2t}\mathbb{V}[Z(t)] \tag{4.40}$$

where μ and $Z(t)$ are defined the same as in (4.32), while $\mathbb{V}[\ ]$ means the second moment of the random variable.

If applying $\mathbb{V}[Z(t)]$ presented by Kelly [67], the equation is changed to the following.

$$eb(\theta, t) = \mu + \frac{\theta}{2}\sigma^2 t^{2H-1} \tag{4.41}$$

where σ^2 is the variance of the random variable and H is the Hurst parameter. If $H \in (0.5, 1)$, the traffic is long range dependent. In such a case,

$$eb(\theta, \infty) = \infty \tag{4.42}$$

Hence, the finiteness hypothesis of (4.14) is breached. From the last section, it is known that even for the long-range dependent traffic, the small buffer behavior is close to exponential queue decay. Therefore, for the large buffer based estimator, all of the traffic cases are assumed as time independent. With such an assumption, the value of H is equal to 0.5 and the time variable t will disappear in the formula. The effective bandwidth is then simplified into the following equation:

$$eb(\theta, t) = \mu + \frac{\theta}{2}\sigma^2 \tag{4.43}$$

From (4.43) it results that $eb(\theta, \infty)$ exists and is finite, $\forall\theta \in \mathbb{R}^+$, iff μ and σ are finite. Therefore this model satisfies the hypothesis of (4.14). For this model, the moment generating function has an analytical solution shown in the following steps.

Replacing (4.43) in (4.15) results in the following:

$$\mu + \frac{\theta\sigma^2}{2} = c \tag{4.44}$$

where c is the bandwidth as introduced in (4.15).

Solving in θ, it results that:

$$\theta^* = \frac{2(c-\mu)}{\sigma^2} \tag{4.45}$$

Replacing θ^* in (4.23) with the value calculated above, one obtains the following expression that links the loss to the main model parameters (μ and σ^2):

$$ln(P_{loss}) = -\frac{2(c-\mu)b}{\sigma^2} - ln(\frac{2\mu(c-\mu)}{\sigma^2}) \tag{4.46}$$

If one uses 10 as the base for the logarithm function, then above expression can be changed to:

$$log(P_{loss}) = -\frac{2b(c-\mu)}{\sigma^2}log(e) - log(\frac{2\mu(c-\mu)}{\sigma^2}) \tag{4.47}$$

Replacing μ and σ^2 by their *estimated* values $\hat{\mu}$ and $\hat{\sigma}$, and using *lbe* to represent the b-asymptote approximation for packet loss probability $log(\hat{P}_{loss})$, the following equation is satisfied:

$$lbe = -\frac{2b(c-\hat{\mu})}{\hat{\sigma}^2}log(e) - log(\frac{2\hat{\mu}(1-\hat{\mu})}{\hat{\sigma}^2}) \tag{4.48}$$

4.2.4 Aggregate Traffic based Estimator

For the general fractional Gaussian Noise model with $H \in [0.5, 1)$ the solution of the constrained optimization problem is obtained at:

$$t^* = \frac{H}{1-H}\frac{b}{c-\mu} \tag{4.49}$$

$$\theta^* = b^{1-2H}\frac{(1-H)^{2H-1}(c-\mu)^{2H}}{H^{2H}\sigma^2} \tag{4.50}$$

By replacing (θ, t) in Equation 4.31 the following expression for the loss probability at a FIFO multiplexer fed by fractional Gaussian Noise is obtained:

$$log(P_{loss}) = -\frac{b^{2-2H}(c-\mu)^{2H}}{2\sigma^2}\frac{(1-H)^{2H-2}}{H^{2H}} - log(\sqrt{2\pi}\frac{b^{2-3H}\mu}{\sigma^3}\frac{(c-\mu)^{3H}H^{-3H}}{(1-H)^{2-3H}}) \tag{4.51}$$

When $H = 0.5$, the solution of the constrained optimization problem of (4.31) is obtained at t^* and θ^*:

$$t^* = \frac{b}{c-\mu} \tag{4.52}$$

$$\theta^* = 2\frac{c-\mu}{\sigma^2} \tag{4.53}$$

The same algebraic process allows us to calculate the loss probability given as follows:

$$\mathbb{P}_{loss} = \frac{e^{\frac{-2b(c-\mu)}{\sigma^2}}}{\frac{2\mu(1-\mu)}{\sigma^2}\sqrt{4\pi\frac{2b(c-\mu)}{\sigma^2}}} \tag{4.54}$$

After applying the logarithm and replacing the estimated values, the following estimator can be derived.

$$ate = -\frac{2b(c-\hat{\mu})}{\hat{\sigma}^2}log(e) - log(\frac{2\hat{\mu}(1-\hat{\mu})}{\hat{\sigma}^2}) - \frac{1}{2}\log(4\pi\frac{2(c-\hat{\mu})b}{\hat{\sigma}^2}) \tag{4.55}$$

It is meaningful to compare the expression of the loss estimator given by (4.55) with the equation given by (4.47). It can be observed that two identical terms are present in both cases. Therefore, the difference between the two estimators yields:

$$\hat{\Delta} = lbe - ate \tag{4.56}$$

$$= \frac{1}{2}\log(4\pi\frac{2(c-\hat{\mu})b}{\hat{\sigma}^2}) \tag{4.57}$$

Hence, the difference between the values estimated by the two estimators is dependent on both the characteristics of the multiplexer and the traffic conditions. The performance of the two estimators may be entirely different in variable contexts.

4.3 Experiment Design and Measurement

This section presents the experiment design and implementation issues to evaluate the performances of the two estimators. All of the measurements are collected from the NCCT & NCIT*net [39].

4.3.1 Testbed Design

Figure 4.3 displays the network topology used in this experimental study. Three BGP/MPLS VPN services are set as services between the two Cisco routers (192.168.2.1 and 192.168.3.1, respectively), where the connection bandwidth is set to 100M/1Gbps. The first VPN is composed by one DVD flow from MAC4 to MAC3 and one background Ethernet traffic from the GN Nettest to MAC3. The second VPN also has a DVD MPEG2 video flow from MAC2 to MAC1 and an Ethernet background traffic flow from GN Nettest to

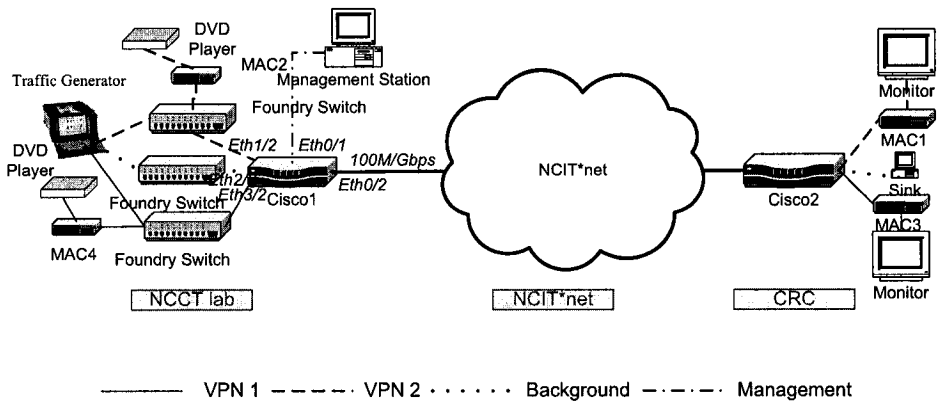


Figure 4.3: Test Bed in NCIT*net2 network

MAC1. VPN3 generates the background traffic, which can simulate many on-off Ethernet traffic flows. All of the VPN services are aggregated on the core link between the two Cisco routers. One management workstation is connected with all of the devices through Interface eth0/1 in the first Cisco router to measure and control the VPN services and traffic. It is used for the definition, creation and activation of the VPN services above, and for all adjacent network management and element management operations. The management traffic is separated from the experimental VPN data traffic through configuring different VLANs in the network.

4.3.2 Traffic Generation

Using a packet traffic generator (see Figure 4.3), different traffic flows can be defined, generated and multiplexed. In the experiments conducted, a traffic generator built by GNNettest, Interwatch 95000, is used. When defining an on-off traffic flow, three parameters need to be specified: packet size, burst length and user utilization. The packet size is variable from 64 to 1500 bytes. The burst length is within 10 to 100 packets. The utilization is the percentage of the total output bandwidth from the generator, which supports 10M/100M/G/10G Ethernet interface.

The MPEG 2 traffic flow is generated by the Multimedia Access Concentrator 500 (MAC 500), which receives the real time DVD images from the DVD player and transcodes to MPEG 2 format digital data. The video traffic is then transmitted using

RTP/RTCP protocol. The traffic rate can be adjusted when initializing the flow; its range in this experiment is set from 1.5M to 50M.

4.3.3 Traffic and Loss Measurement

Traditional traffic measurement usually measures traffic through polling network element interfaces, but it cannot provide detail information about the usage for different users and flows which aggregate into one link in the high speed network, because the MIB counters do not have the flow information. Furthermore, polling the interface through SNMP will cause traffic and decrease the performance of the network. Therefore, in this work SNMP is not used to measure traffic; it is only employed to measure the packet loss, since packet loss measurement will run only once for each experiment.

Trace based traffic monitor, such as TCPDump, can provide more detailed information. However, usually it is based on Ethernet packet capturing, and cannot be used widely for monitoring a whole network. Another problem is that huge traffic data will be generated for the high speed network causing the monitor inapplicable for the high speed network.

Thus, in this thesis, flow level traffic measurement technology is employed, which brings information less complete than the one obtained by packet level measurements; however, this technology is sufficient for our applications. Currently, sFlow and netFlow [86] are the most popular protocols.

sFlow technology is configured on the Foundry switch to generate the traffic statistic data. The measurement station is connected to the Foundry Switch to get the traffic statistics through UDP/sFlow protocol. NetFlow is a traffic monitoring technology developed by Cisco Networks and is configured on the Cisco routers in the testbed. The minimal interval time for netFlow agent to export data is set to 1 second. The implemented software receives and analyzes the traffic statistic data in the PDU, then calculates the aggregate traffic. The software architecture is shown in Figure 3.8. The implementation is detailed and demonstrated in Chapter 3. Three proxies designed in Java are used to collect the different traffic statistics from the devices. The collected information is analyzed by the analyzer and finally the required data are stored in the DB and also displayed on the monitor.

The total traffic in each interval is measured by the sum of all VPN flow traffic. It is expressed as:

$$\hat{\alpha}(k) = \sum_{i=1}^{N} \hat{\alpha}_i(k) \tag{4.58}$$

where $\hat{\alpha}(k)$ is the aggregate input traffic to the P router in the k^{th} time interval; and $\hat{\alpha}_i(k)$ is the output traffic from VPN source i. In this testbed, $N = 3$.

To estimate the packet loss, the traffic mean $\hat{\mu}$ and variance $\hat{\sigma}$ should be calculated based on the measured traffic in each time interval $\hat{\alpha}(k)$. They can be calculated as:

$$\hat{\mu} = \frac{1}{N} \sum_{k=1}^{N} \hat{\alpha}(k) \tag{4.59}$$

$$\hat{\sigma}^2 = \frac{1}{N-1} \sum_{i=1}^{N} [\hat{\alpha}(k) - \hat{\mu}]^2 \tag{4.60}$$

where N is the size of the time periods for the each experiment, and $\hat{\alpha}(k)$ is the measured traffic in the k^{th} time interval.

Packet loss ratio (plr) is used as the performance evaluation criterion. In this test, to measure *plr* of the aggregate traffic in the core link, interface counters in MIB II [79] is polled through SNMP protocol after each experiment. The *plr* can be calculated by the following equation:

$$plr = \frac{N_l}{N_l + N_O} \tag{4.61}$$

where N_l means the total packet loss number. N_O is the transported packets. Both can be retrieved from the interface counters in standard MIB II from the first Cisco router.

4.4 Numerical Result Analysis

This section compares and analyzes the performance of two estimators introduced in the previous sections. The values of loss probability predicted by the estimators are compared with the observed loss ratio values in a series of experiments.

4.4.1 Aggregated Traffic Model

The Central Limit Theorem ensures that by the aggregation of a large number of independent traffic streams the resulting aggregated traffic process will be a Gaussian

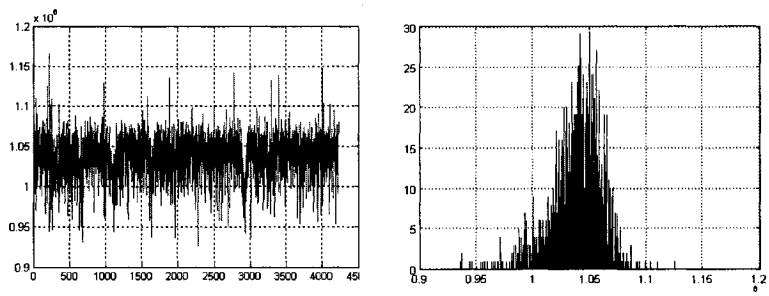


Figure 4.4: 20-Flow Aggregation Traffic Measured on Eth1/2 of Cisco1

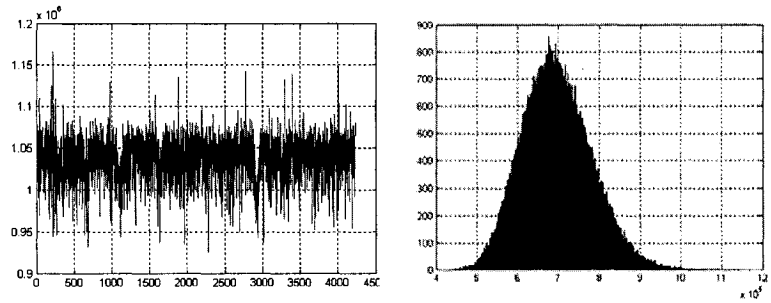


Figure 4.5: 40-Flow Aggregation Traffic Measured on Eth1/2 of Cisco1

process. Therefore, the first test to be performed will be directed toward the verification of the above statement. Furthermore, we will also try to adjust the parameters of the traffic generator to make the generated traffic satisfy the assumptions.

For this goal, we firstly generate 20 on-off flows with different flow parameters. Then 2 DVD flows with 1.5M rate and 2M rate are mixed together. The aggregate traffic on the core link is measured. Step by step, the number of on-off flows is increased to 40 and the aggregated traffic is also recorded.

The Chi-square (χ^2) test [111] is used to verify if the traffic data was generated by a stochastic process with a Gaussian distribution. The test requires that the data first be grouped in different bins. The actual number of observations in each group is compared to the expected number of observations and the test statistic is calculated as a function of this difference. The Chi-square test is applied depending on which of the following hypothesis is true:

- H_0: The data follows a specified distribution, which in this case is the Gaussian

distribution.

- H_a: The data does not follow the specified distribution.

For the Chi-square goodness-of-fit computation, the data are divided into k bins and the test statistic is defined as:

$$\chi^2 = \sum_{i=1}^{k} (O_i - E_i)^2 / E_i \tag{4.62}$$

where O_i is the observed frequency for bin i and E_i is the expected frequency for bin i. The expected frequency is calculated by:

$$E_i = N(F(Y_u) - F(Y_l)) \tag{4.63}$$

where F is the Cumulative Distribution Function (CDF) for the distribution being tested, Y_u is the upper limit for the class i, Y_l is the lower limit for class i, and N is the sample size.

The test statistic of the core traffic follows, approximately, a Chi-square distribution with $(k - c)$ degrees of freedom where k is the number of non-empty cells and c is the number of estimated parameters for the distribution plus 1.

Therefore, the hypothesis that the data are from a population with the specified distribution is rejected if

$$\chi^2 > \chi^2_{(\alpha, k-c)} \tag{4.64}$$

where α is the significance level, χ^2_α is the upper critical value and $\chi^2_{1-\alpha}$ is the lower critical value from the Chi-square distribution.

Test results for 20-flow and 40-flow traffic data are shown in Table 4.1 and 4.2 respectively, which proves that the core traffic has a Gaussian distribution when it is composed of a lot of flows.

From the traffic plot and related distribution shown in Figure 4.4 and 4.5, it can be visually verified that the total traffic is more likely Gaussian traffic with the increasing number of flows. This type of traffic will be used in the following experiments.

4.4.2 Single VPN Loss Vs Aggregated Traffic loss

Service guarantees should be provided for each single VPN service. However, in the practical networks, only aggregated traffic information can be available. (4.39) assumes

Number of Observations	4000
Number of Non-empty Cells	24
Chi-square Test Statistic	2030.25
Degrees of Freedom	23
Chi-square CDF Value	1.00
α Level	0.01
Conclusion	Reject H_o

Table 4.1: Chi-square test results for 20-flow traffic data

Number of Observations	4000
Number of Non-empty Cells	24
Chi-square Test Statistic	18.25
Degrees of Freedom	23
Chi-square CDF Value	0.271
α Level	0.01
Conclusion	Accept H_o

Table 4.2: Chi-square test results for 40-flow traffic data

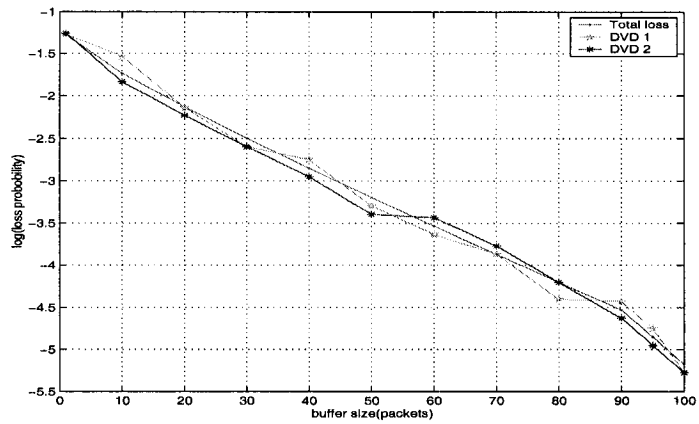
that the loss experienced by the individual VPN service should be equal to the loss of the aggregate traffic.

In the following experiments, the loss ratios of the two DVD flows and the aggregated traffic are measured and showed in Figure 4.6. From Figure 4.6, one can see that the values of the loss probability for the two DVD flows and aggregate flow are very close to each other, although the individual probability loss has a more significant variation than the aggregate loss.

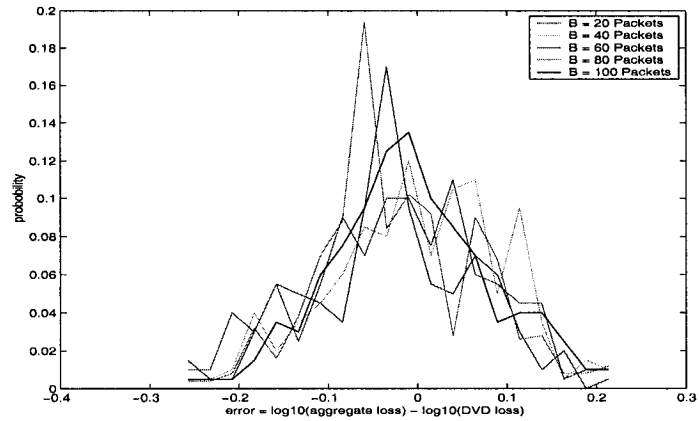
To make the comparison more complete and accurate, the error between the aggregate loss and one DVD flow loss is recorded while changing the utility and buffer size in an appropriate range. Figure 4.6 shows the PDFs of the results during a large number of tests. The results show that the difference between the individual loss and aggregate loss is very small. Therefore, it can be stated that the aggregate-flow loss probability can represent the individual-flow loss probability with a high degree of accuracy.

4.4.3 Performance Evaluation of Two Estimators

To compare the performance of the two estimators, the error for each experiment $\hat{\epsilon}_k$, between the measured loss ratio $log(plr)$ and the packet loss probability $log(p_{loss})$ es-



Sample Path on Packet Loss Probability



PDF for the Error of Packet loss probability

Figure 4.6: Error Between Individual-Flow Loss and Aggregated-Flow Loss

timated by the estimator for each experiment, is calculated according to the following equation:

$$\hat{\epsilon}_k = log(p_{loss}) - log(plr) \tag{4.65}$$

where, $log(p_{loss}) = lbe$ or ate. lbe is the loss estimator based on the large buffer and ate is the loss estimator based on the aggregate traffic.

Then, the following quantitative criteria for the errors are calculated for each buffer size: Error Mean, Error Variance, Min Error, Max Error and Error Range.

- Error Mean. This criterion is the sample mean of each of the estimators at each of the buffer sizes.

$$\bar{\varepsilon} := \frac{1}{N}\sum_{k=1}^{N}\hat{\varepsilon}_k$$

Buffer (Packets)	Error Mean	Error Variance	Min Error	Max Error	Error Range
20	1.7735	0.0561	1.5615	2.5942	1.0328
40	1.6715	0.0413	1.4720	2.5134	1.0414
60	1.5669	0.0413	1.3500	2.4292	1.0792
80	1.4594	0.0576	1.1907	2.3402	1.1495
100	1.3493	0.0897	0.8197	2.2405	1.4209
120	1.2358	0.1371	0.3992	2.1511	1.7519
140	1.1182	0.1986	-0.0194	2.0503	2.0697
160	0.9973	0.2741	-0.4423	1.9662	2.4086
180	0.8749	0.3632	-0.8637	1.9187	2.7824
200	0.7533	0.4712	-1.2834	1.9380	3.2213

Table 4.3: lbe Statistics Synopsis on Measured Traffic and Loss on Cisco1

- Error Variance. This metric is defined by the following equation:

$$\sigma^2(\varepsilon) := \frac{1}{N-1}(\sum_{k=1}^{N}(\hat{\varepsilon}_k - \bar{\varepsilon})^2)$$

- Min Error. It is the minimum error value from the dataset.

$$\varepsilon_{min} := min\{\hat{\varepsilon}_k \mid k \in [1, N]\}$$

- Max Error. It is the maximum error value from the dataset.

$$\varepsilon_{max} := max\{\hat{\varepsilon}_k \mid k \in [1, N]\}$$

- Error Range. It is defined as the difference between the maximum value, from a given data set, and the minimum value from the same set.

$$\delta := \varepsilon_{max} - \varepsilon_{min}$$

The above criteria are computed for all data sets obtained from experiments through changing the link utility from 70% to 90%, the packet size from 64 byte to 1200 bytes, and the weight of the background traffic from 50% to 90%. Each experiment is continually run for 30 minutes. Then the lost packets and total packets are collected from the MIB II. The traffic mean for each time period (1 second) is collected from netFlow output.

Table 4.3 summarizes the statistics for *lbe* estimator. The comparison of the estimated loss values with the results of measurements shows that the Error Mean is about

Buffer (Packets)	Error Mean	Error Variance	Min Error	Max Error	Error Range
20	1.4612	0.0325	1.2588	2.1315	0.8727
40	1.2087	0.0336	1.0019	1.9001	0.8982
60	1.0161	0.0497	0.7475	1.7279	0.9804
80	0.8461	0.0822	0.3624	1.5764	1.2141
100	0.6875	0.1305	-0.0861	1.4283	1.5144
120	0.5344	0.1942	-0.5461	1.2992	1.8454
140	0.3834	0.2723	-0.9982	1.178	2.1762
160	0.2335	0.3645	-1.4502	1.1241	2.5742
180	0.0856	0.4698	-1.8971	1.0733	2.9704
200	-0.059	0.593	-2.3396	1.0262	3.3658

Table 4.4: ate Statistics Synopsis on Measured Traffic and Loss on Cisco1

1 or 2 degrees of order. Another important property of the *lbe* estimator is that it almost has a positive error bias. Note that a positive bias is more useful than the negative one. This is due to the asymmetry of consequences in the two cases: if the loss is overestimated then the overall result is that the system will operate at a lower utilization. The decrease in utilization is not generally dramatic. On the other hand, the violation of the loss targets usually has more negative economic consequences either because of possible penalties or because it might be severe on the quality of experience (QOE) in regards to services.

The results for the *ate* estimator are aggregated in Table 4.4, which shows that the error bias is variable. While the bias is positive for the small buffer multiplexer, it is negative for larger buffer sizes. However, the Error Mean is significantly smaller than the case of *lbe*.

By comparing the errors between *lbe* and *ate*, *lbe* mostly has a positive bias, which means that in most of the simulated cases *lbe* overestimates the loss at the multiplexer. Since the actual loss value experienced at the link must be lower than a preset value, the unidirectional bias is useful. The effect of overestimating loss is a decrease in the link utilization. The decrease is usually minute, since at high link capacities, because of the high degree of multiplexing, even decreases of 1% in utilization results in decreases of one order of magnitude in loss.

Therefore, if the loss is estimated in the open loop system and the provider cannot break the SLA; *lbe* is more useful than the *ate*.

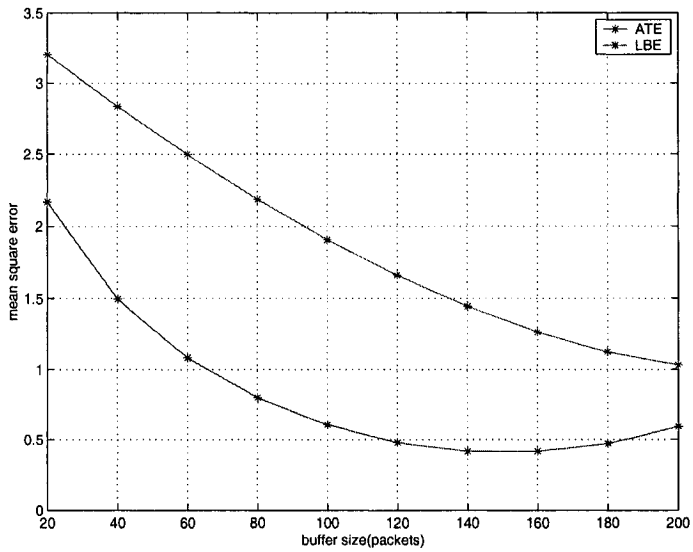


Figure 4.7: Comparison of Mean Square Errors

Mean square Error (MSE) is simply the average of the squared individual error values from each set of data:

$$mse = \frac{1}{n}(\sum_{k=1}^{n}(\hat{\epsilon}_k)^2)$$

Through Figure 4.7, MSE of *ate* method is smaller than *lbe* method. Therefore, *ate* estimator has better performance than *lbe* method. However, *ate* has non-constant biases, and it will more easily cause the QoS to break. Therefore, if the loss guarantee is not very strict, i.e. the provider can break the contract, the *ate* provides more utilization.

4.4.4 Estimation Error Analysis

Both of the estimators have shown error variance in a certain range. In this section, we will try to find which factors will affect these errors.

Traffic Model

The traffic arrival process for each VPN service is assumed to be Gaussian; the packet loss probability is calculated using the Moment Generating Function for this model. However, the real traffic may not have the exact Gaussian traffic characteristics. If the non Gaussian traffic is in case, there are probably some biased errors. To verify if this type of error exists, the non-Gaussian traffic, similar Gaussian traffic and Gaussian

traffic are generated as the input traffic, and the loss curve is compared among them.

First, one on-off flow is generated from the generator with the parameters specified in Table 4.5. The loss curve for different buffer size is shown in Figure 4.8 (I) as On-Off model.

Table 4.5: Parameter Configuration for the Traffic Generator

Parameter	value
Packet size	1498 byte
On period	20 ms
Off period	40 ms
Bandwidth	6.7Mbps
Peak rate	20Mbps

Then 5 on-off flows with a completely different on-off period are aggregated. In this case, the traffic should be similar to the Gaussian model. The loss curve is shown in Figure 4.8 (II).

Finally, 40 on-off flows with the different on-off period and different packet size are merged with another 2 DVD flows. The aggregation output is a Gaussian traffic as proved in Figure 4.5. The different estimation values have been shown as Figure 4.8 (III).

From these figures, the character of loss curves is evidently different. When the traffic is more likely Gaussian, the error is becoming more positive and constant. This fact is observed on the majority of experiments and tests collaborated for the purpose of the research presented in this thesis. However, the traffic like one-flow and five-flow, can also happen to be present in the network, making the error sometimes negative, especially for the *ate* method.

Buffer Size

Since the *lbe* is based on large buffer asymptotic, it is possible that the process behavior at large buffers is completely different from the behavior in the low buffer range. However in the test presented, the buffer size is not too large because the practical application limits the size of buffer. Experiments are designed to test if the error will be affected by the different range of buffer sizes, where three types of traffic are generated: On-off

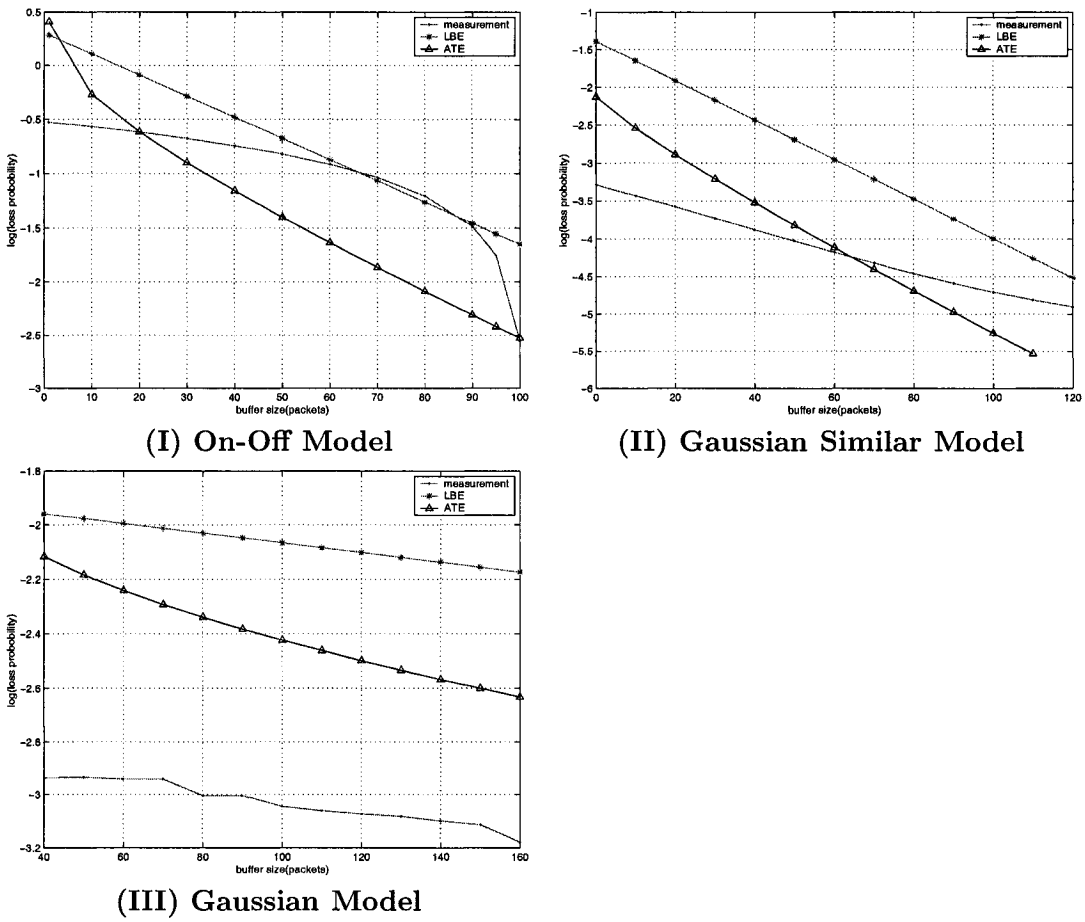


Figure 4.8: Estimation with Different Input Traffic Models

Model traffic, Gaussian similar traffic and Gaussian traffic. The different type of traffic are generated by the traffic generator and measured on the Cisco1. The results are shown in Figure 4.9. Appearing in the figure, the loss curve is likely composed with three fold lines, when the buffer size is increased from zero to a large number.

1. The first part of the measured loss curve, especially from buffer size 20 to 160, is a line which has almost the same slope as the estimation curves. The similarity of the three lines means the packet loss is exponentially decreasing with increases in buffer size. In this range of buffer sizes, the gap between the lines (i.e., the estimation error) is almost constant.

2. The second part of the measured loss curve is very flat, which is mostly in the buffer range [200, 600]. The flatness means the packet loss will not change significantly

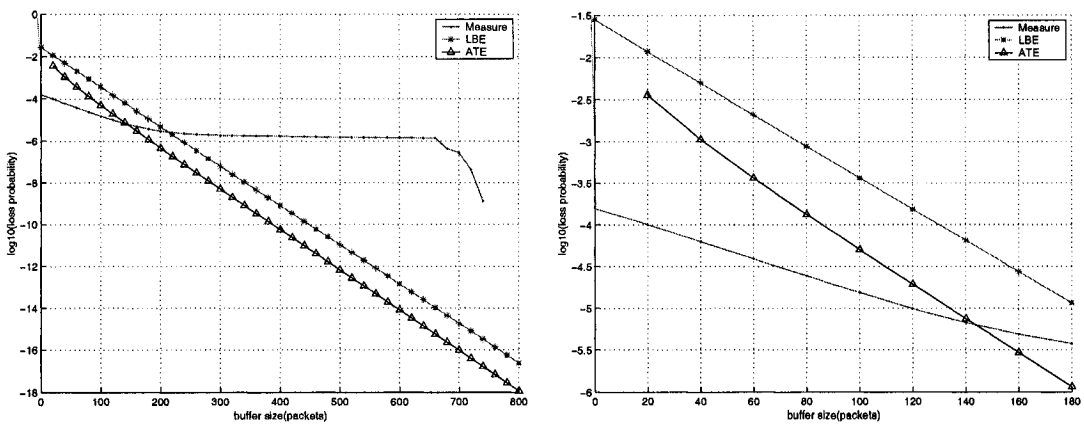


Figure 4.9: Estimation with Different Buffer Range

with the increase in buffer size. In such a case, the loss is probably caused by the large burst traffic, which will not be affected significantly by increasing the buffer size.

3. The third part is a steep line if the buffer size is over 600 packets. The reason is because the loss will sharply decrease or disappear when the buffer size is large enough to contain all the burst data.

The length for each line phase in the measured loss curve depends on the length of the burst data and the variance of the traffic model. For the traffic generated in the experiments reported, the measured loss curve almost occurs on the first linear portion, which has the similar slope as the loss estimation curve.

4.5 Conclusion

This chapter tries to derive two model based loss estimators for practical application. Both of the packet loss estimators are evaluated in the conditions of a live network with real traffic. The results show that the proposed estimators can be employed effectively to calculate the real packet loss probability. On the other hand, the analysis of the estimation error also indicates that there is still enough room to improve the estimators, which will be studied in the next chapter.

Chapter 5

Online Estimation and Preliminary Control

In the preceding chapter, two types of packet loss estimators have been studied in the practical MPLS VPN networks: the Large Buffer asymptotic Estimator (lbe) and the Aggregate Traffic approximation Estimator (ate). The experimental results show that there is enough room for improving the estimator errors. In this chapter, a simple Reactive Estimator (re) is constructed on-line, which extends the original *lbe* estimator with the capacity of adapting itself to the variable contexts. Specifically, *re* will be automatically adjusted by employing one dynamic item based on the feedback of loss ratio measurement, so that it estimates the packet loss probability more accurately within the time-varying traffic model. A series of experiments are devised to evaluate the performance of the new estimator under different traffic arrivals and different buffer sizes. The new on-line estimator is applied to the service admission control. The classical PI control technology is studied for the analytic model to see whether the preliminary controller can address the packet loss issue.

5.1 Online Estimation

5.1.1 Online Measurement, Estimation and Control

The packet loss control architecture for VPN services has been proposed in the Chapter 3, where the estimator is employed to calculate the packet loss probability based on the measuring of the flow traffic. In Chapter 4, two estimators are derived and evaluated based on the measured offline traffic data: one is the Large Buffer Based Estimator (*lbe*) and the other is the Aggregate Traffic asymptotic Estimator (ate).

When the estimators are utilized in the practical application, such as the dynamic loss control system, the packet loss probability should be calculated on line. That is, to control the network with a timely response, if the system is modeled as a discrete time system, the packet loss probability should be estimated for each time interval.

Let's assume that, using the same notations as in Chapter 4. Section 2, the discrete time interval is noted by T, which means that the packet loss probability is estimated every T unit. Let $lbe(k)$ denote the estimator for estimating the loss probability in the k^{th} time interval $[kT, kT+T)$, $k \in N$. Then the b-asymptote estimator lbe in (4.47) will be changed to the following equation.

$$\begin{aligned} lbe(k) = &- \frac{2b(c-\hat{\mu}(k))}{\hat{\sigma}^2(k)} log(e) \\ &- log(\frac{2\hat{\mu}(k)(1-\hat{\mu}(k))}{\hat{\sigma}^2(k)}) \end{aligned} \tag{5.1}$$

where $\hat{\mu}(k)$ and $\hat{\sigma}(k)$ are estimated traffic mean and variance for the same k^{th} time interval, respectively. They can be calculated as the following:

$$\hat{\mu}(k) = \frac{1}{N} \sum_{i=0}^{N-1} \hat{\alpha}(k-i) \tag{5.2}$$

$$\hat{\sigma}^2(k) = \frac{1}{N-1} \sum_{i=0}^{N-1} [\hat{\alpha}(k-i) - \hat{\mu}(k)]^2 \tag{5.3}$$

where N is the size of the moving window for calculating the average of the traffic mean and variance, and $\hat{\alpha}(k)$ is the measured traffic in the k^{th} time interval.

In the similar way, the online estimator ate is formulated as

$$\begin{aligned} ate(k) = &- \frac{2b(c-\hat{\mu}(k))}{\hat{\sigma}(k)^2} log(e) - log(\frac{2\hat{\mu}(k)(1-\hat{\mu}(k))}{\hat{\sigma}(k)^2}) \\ &- \frac{1}{2} \log(4\pi \frac{2(c-\hat{\mu}(k))b}{\hat{\sigma}(k)^2}) \end{aligned} \tag{5.4}$$

5.1.2 Reactive Estimator

The performance evaluation of the two estimators in last chapter shows that the mean error of ate is less than lbe, but its error variation is much larger. That is, the error between the lbe and the actual packet loss probability is more stable.

In the error analysis, one can find that if the traffic configuration is known and unchanged, the error is almost kept as the same value. The result encourages us to

construct a reactive estimator, which will adjust the error according to the variable context. Since *lbe* has a smaller error variation, it is selected as the better base candidate in the work.

The assumptions made about the input traffic are crucial for the estimator, so that the asymptotic results are different for long range-dependent traffic and short-range dependent Gaussian traffic. In order to improve the accuracy and to maintain applicability when used in a live network, a simple Reactive Estimator (RE), which can be adaptive to the dynamic traffic model, is constructed as the followings.

Let assume, the $plr(k)$, the k^{th} real-time measured pack loss ratio can be available according to the following measurement and calculation.

$$plr(k) = log(\frac{\sum_{i=0}^{R-1} \hat{l}(k-i)}{\sum_{i=0}^{R-1} \hat{\alpha}(k-i)}) \tag{5.5}$$

where $\hat{l}(k)$ is the number of lost packets measured in the k^{th} time interval and $\hat{\alpha}(k)$ is the number of total input packets measured in the same time interval. Both of them can be retrieved from the interface counters in standard MIB II. R is the size of the moving window for calculating the packet loss rate. To make the calculation accurately, the R should be large enough.

Based on the basic b-asymptote estimator, a simple reactive estimator $re(k)$ for k^{th} time interval is constructed in (5.6).

$$re(k) = lbe(k) + di(k) \tag{5.6}$$

where, $lbe(k)$ is calculated with (5.1) and $di(k)$ is a dynamic item, added to adjust the error.

A linear prediction technique [77] is used to calculate the dynamic item with the following equation.

$$di(k) = \sum_{l=1}^{p} [plr(k-l) - re(k-l)]w(l) \tag{5.7}$$

where $w(l)s$ are the linear prediction coefficients to define or calculate so that the error due to prediction can be minimized, and p is the size of the moving window.

In the thesis, a simple prediction scheme is used since practical applications need minimal calculation complexity. In the analysis and following test, we use moving window average to calculate the predicted dynamic item. In moving average with window

size of p, linear prediction coefficients are:

$$w(i) = \frac{1}{p}, \forall i \in [1, p] \tag{5.8}$$

Therefore, the $di(k)$ is simplified into the following equation.

$$di(k) = \frac{1}{p}\sum_{l=1}^{p}(plr(k-l) - re(k-l)) \tag{5.9}$$

$$= di(k-1) + \frac{1}{p}(plr(k-1) - re(k-1) - plr(k-1-p) + re(k-1-p)) \tag{5.10}$$

The RE can be calculated as:

$$re(k) = lbe(k) + \frac{1}{p}\sum_{l=1}^{p}(plr(k-l) - re(k-l)) \tag{5.11}$$

After z-transform, the RE can be written as:

$$re(z) = \frac{lbe(z)p(1-z^{-1}) + plr(z)(z^{-1} - z^{-p})}{p + (1-p)z^{-1} - z^{-p-1}} \tag{5.12}$$

When the estimator is implemented on line, it is calculated by (5.1) and (5.10). The dynamic item can be calculated recursively, thus it is easier to accomplish in the practical implementation.

5.1.3 Kalman Filter on the Mean and Variance

The calculation of the estimator is largely dependent on the packets arrival mean and variance. Therefore, one key issue is to provide the best estimate of the mean, denoted by μ_k, and the variance, denoted by σ_k, of the instant packet rate of the input traffic process [38]. In this section, these two variables are assumed as independent and optimally estimated by the Kalman filter.

The state of the system is defined as $X_k = [\mu_k, \sigma_k]^T$. The dynamics of the mean and variance system model is defined as [53]:

$$X_k = AX_{k-1} + Bu_k + w_k \tag{5.13}$$

where u_k is the instant packet rate input. The noise, $w_k = [w_k^m, w_k^v]^T$, is white noise source with zero mean and covariance matrix Q and is uncorrelated with the input. A and B are constant matrixes defined as:

$$A = \begin{bmatrix} 1 - 1/N & 0 \\ 0 & 1 \end{bmatrix} B = \begin{bmatrix} 1/N & 0 \\ 0 & 0 \end{bmatrix}$$

where N is the sample size.

In Dziong's scheme [38] the parameters are simple, where the mean is assumed the same as the previous one. Therefore, their parameter matrix is defined as:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

In the network, the packet rate mean and variance can be measured and expressed by $Z_k = [\bar{\mu}_k, \bar{\sigma}_k]^T$. Therefore, the measurement model is described by

$$Z_k = HX_k + v_k \tag{5.14}$$

where $H = I$ is the constant gain and $v_k = [v_k^m, v_k^v]$ is the measurement noise. The noise, v_k, is also white noise source with zero mean and covariance R that is uncorrelated with the input. The two noise sources are independent of each other and independent of the input.

Given the system depicted above, a Kalman filter is applied to optimally recursively estimate the μ_k and σ_k. The task of the Kalman filter is to filter Z_k so as to estimate the variable X_k while minimizing the effects of w and v.

Kalman filter is a general method for the optimal estimation of a noisy measurement. The basic idea is to use the estimation error obtained from the past measurement of the variables and the estimation equation to fix the one-step prediction. The algorithm runs recursively and is applicable for the on-line application. The system model and Kalman filter approach are shown in Figure 5.1.

The upper part of Figure 5.1 describes the system model, that is specified by equation (5.13) and (5.14). The lower part is used to implement the Kalman filter.

First, the variable, $\hat{X}_k^-$, refers to the *apriori* estimate. It is defined as:

$$\hat{X}_k^- = A\hat{X}_{k-1} + Bu_k \tag{5.15}$$

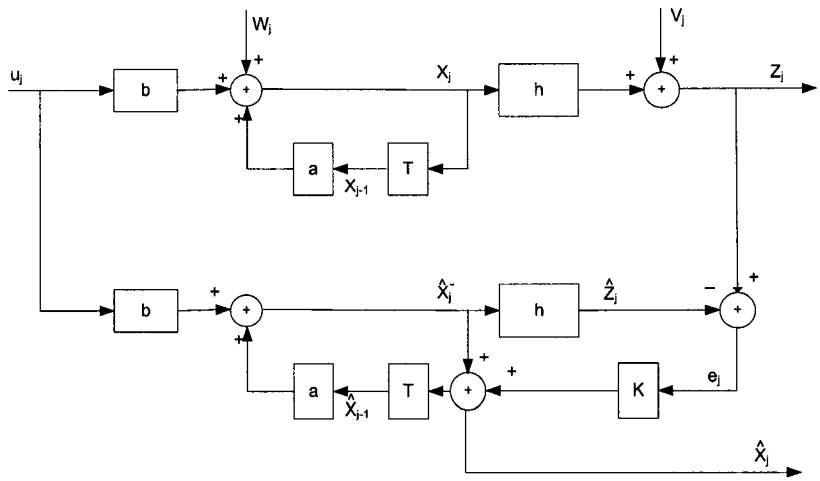


Figure 5.1: System Model and Kalman Filter

This *apriori* estimate is used to predict an estimate for the output, $\hat{Z}_k$. The difference between this estimated output and the actual measured output is called residual. The residual is denoted by e_k and expressed as follows:

$$e_k = Z_k - \hat{Z}_k \tag{5.16}$$

$$= Z_k - H\hat{X}_k^- \tag{5.17}$$

If the residual is small, it generally means a good estimate; if it is large the estimate is not so good. This information is then employed to refine the estimate of X_k. This new estimate is called the *aposteriori* estimate, denoted as $\hat{X}_k$.

$$\hat{X}_k = \hat{X}_k^- + Ke_k \tag{5.18}$$

$$= \hat{X}_k^- + K(Z_k - h\hat{X}_k^-) \tag{5.19}$$

where K is called Kalman gain and the calculation of K is the heart of Kalman filtering.

In order to solve K, two mean squared errors are defined. One is called *apriori* covariance, P_k^-, described as:

$$P_j^- = E(X_k - \hat{X}_k^-)^2 \tag{5.20}$$

The other one is called *aposteriori* covariance and defined as:

$$P_j = E(X_k - \hat{X}_k)^2 \tag{5.21}$$

After calculation, the two covariances are simplified as:

$$P_k^- = AP_{k-1}A^T + Q \tag{5.22}$$

$$P_k = (I - K_k H)P_k^- \tag{5.23}$$

A Kalman filter minimizes the *aposteriori* variance, P_k, by choosing the suitable value of K. Therefore, K is calculated by:

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R} \tag{5.24}$$

$$= \frac{(AP_{k-1}A^T + Q)H^T}{H(AP_{k-1}A^T + Q)H^T + R} \tag{5.25}$$

Based on the above equations, the recursive Kalman filter algorithm is depicted in the following steps:

1. Setup the initial values for Q, R, P_0 and $\hat{X}_0$.
2. Project the *apriori* estimate with equation (5.15)
3. Project the error covariance for the *apriori* estimate with equation (5.22).
4. Compute the Kalman gain with equation (5.25)
5. Update the estimate with measurement with equation (5.19)
6. Update the error covariance for the *aposteriori* estimate with equation (5.23)
7. goto step 2.

5.1.4 Performance Evaluation

In this section, a large number of different flows are designed and mixed to check the accuracy and applicability of the reactive estimator. The performance of the new estimator is compared with the original *lbe* estimator in the live network.

Kalman Filter

Figure 5.2 shows one example of the estimate and measurement of the packet rate mean and Figure 5.3 shows the estimate and measurement of the packet rate variance. The actual value is implemented with the traffic generator and the measured value is obtained from the netflow collector. In both figures, the estimated variables with the Kalman filter approach are much closer to the actual values.

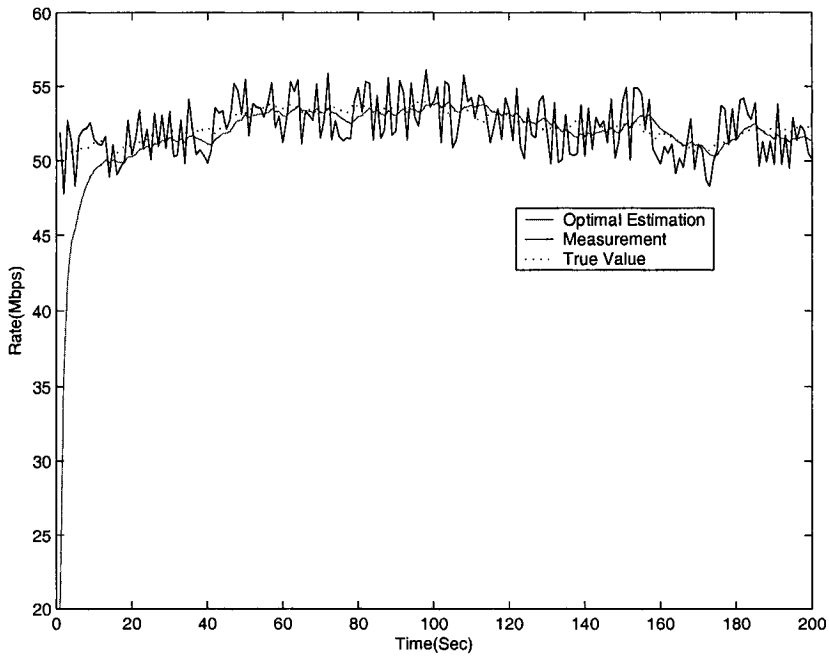


Figure 5.2: Estimate and Measurement of the Packet Rate Mean on the Eth1/2 of Cisco1 in NCCT Network

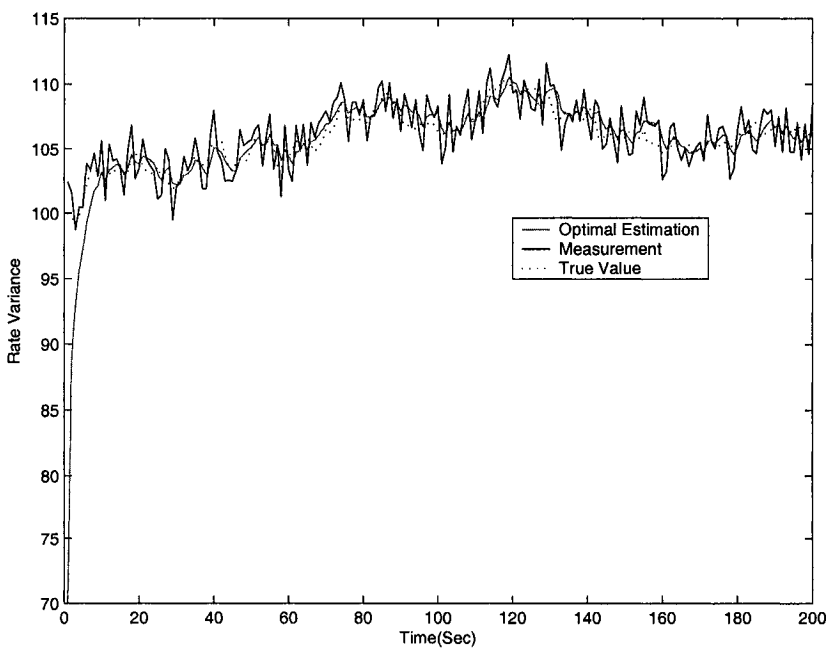


Figure 5.3: Estimate and Measurement of the Packet Rate Variance on the Eth1/2 of Cisco1 in NCCT Network

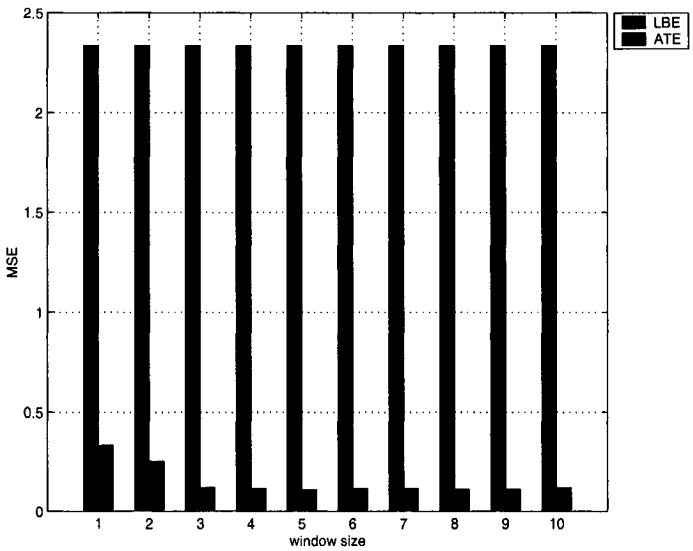


Figure 5.4: MSE vs Window Size

Window Size

One important parameter to affect the performance is the size of the moving window, denoted as p. The size of the window is variable from 1 to 10 in the experiment. Mean Square Error (MSE) is selected as the criterion to decide the best value for p, denoted as p^*. That is, the p^* should be calculated as the following equation:

$$p^* \in \{p \| 1 \leq p \leq 10\}, \ \textit{where } MSE(p^*) \textit{ is optimized.} \tag{5.26}$$

MSE is the average of the squared errors in each test, which is widely used as a standard in the literature. It is defined as:

$$MSE = \frac{1}{n}(\sum_{k=1}^{n}[re(k) - plr(k)]^2) \tag{5.27}$$

The result is shown in Figure 5.4, where it shows that the increasing of window size p gives a better performance. However, after number 3, it does not give a significant better result in terms of the MSE criteria. Using a big window size will increase the complexity of the implementation and will also incur scalability issues. Furthermore, A big window size p is not appropriate for implementing the estimator of the loss probability in an application. So in the next experiments, $p^* = 3$ is used.

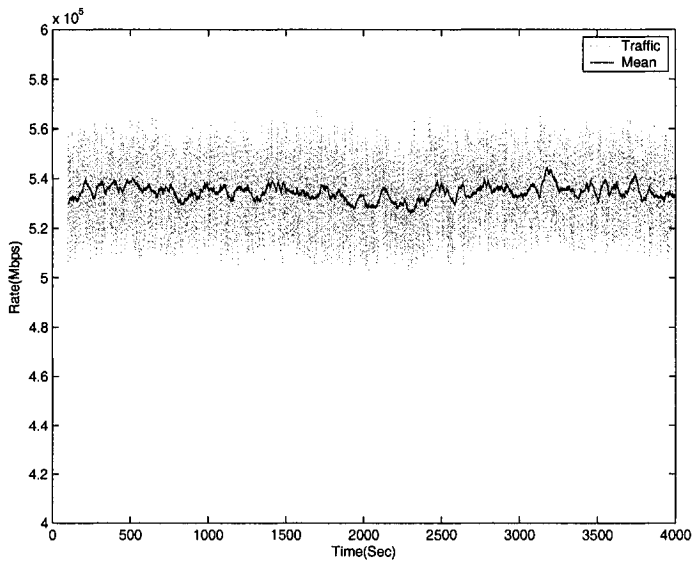
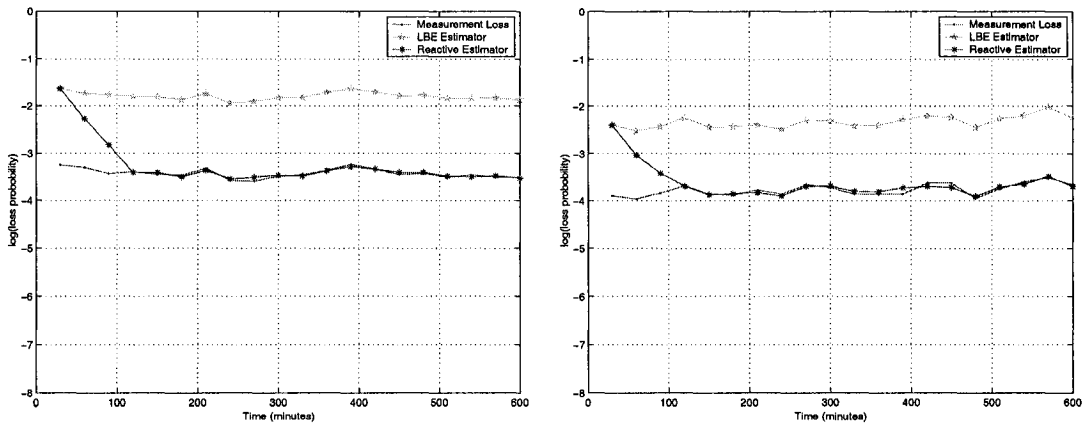


Figure 5.5: Traffic Measurement on the Eth1/2 of Cisco1 in NCCT Network

Stationary Traffic

Firstly, the stationary traffic is generated to examine the performance of the estimator. In order to produce the stationary traffic on the core link, 40 Ethernet flows are output from the packet generator, which are then aggregated with two continually replayed DVD flows. The DVD flows take 20 percents of the total bandwidth and the utilization of the link is 90 percents. The measured traffic and the mean of the traffic are shown in Figure 5.5. From the figure, the mean of the traffic is almost constant with time changing. Moreover, the variance does not vary too much with the time, which is also indicated by the shape of the traffic. In such a way, the aggregated traffic is similar with the stationary Gaussian traffic.

With the moving window size p set to 3, Figure 5.6 (I) and (II) show the actual measured loss rate, *lbe* estimate and the reactive estimate in one of the experiments. *lbe* estimate is closely related with the actual loss, but the error is about one or two orders of magnitude. For the reactive estimator, in the beginning, when the dynamic item for error adaptation is not setup, the reactive estimate shows the same error as *lbe* estimate. However, once the dynamic item is achieved after a number of intervals (window sizes), the reactive estimate converges to the actual loss very tightly. When buffer size increases from 60 packets to 120 packets, the loss probability decreases from



Loss Probability $= 10^{-3.5}$ **Loss Probability** $= 10^{-4}$

Figure 5.6: Packet Loss Time Series for Measured plr, lbe and re Based on the Traffic from the Eth1/2 of Cisco1 in NCCT Network

$10^{-3.5}$ to 10^{-4}. The estimators also decrease almost the same degree. The results show that the adaptive-reactive estimator follows the real measured loss very closely, demonstrating its effectiveness.

Different buffer sizes may result in different effects on the effectiveness of the estimator. To evaluate if there are different performances with variable buffer sizes, More than 100 experiments were performed, in which the buffer size is maintained variable from 20 packets to 200 packets while the aggregate traffic maintains the same. The errors between the measured loss ratio and the loss estimate were recorded as $\epsilon(lbe)$ and $\epsilon(re)$, respectively.

$$\epsilon(lbe) = lbe - plr \tag{5.28}$$

$$\epsilon(re) = re - plr \tag{5.29}$$

The following statistic criteria were employed on the error: Mean, Variance, Min, Max and Range. The results of comparing the loss predicted by the new reactive estimator with the measured loss ratio are aggregated in Table 5.1. It shows that all the error statistics are better than lbe that is shown in Table 5.2. Especially, the error mean from 0.0011 to 0.0029 is significantly smaller than the one in the case of lbe that ranges from 0.7533 to 1.7735.

To see clearly the distribution of the errors, three PDFs of the errors are given with buffer size = 40, 80 and 120 packets in Figure 5.7. It shows that although the

Buffer (Packets)	Error Mean	Error Variance	Min Error	Max Error	Error Range
20	1.7735	0.0561	1.5615	2.5942	1.0328
40	1.6715	0.0413	1.4720	2.5134	1.0414
60	1.5669	0.0413	1.3500	2.4292	1.0792
80	1.4594	0.0576	1.1907	2.3402	1.1495
100	1.3493	0.0897	0.8197	2.2405	1.4209
120	1.2358	0.1371	0.3992	2.1511	1.7519
140	1.1182	0.1986	-0.0194	2.0503	2.0697
160	0.9973	0.2741	-0.4423	1.9662	2.4086
180	0.8749	0.3632	-0.8637	1.9187	2.7824
200	0.7533	0.4712	-1.2834	1.9380	3.2213

Table 5.1: Statistics of $\epsilon(lbe)$ for Stationary Traffic Measured from the Eth1/2 of Cisco1 in NCCT Network

Buffer (Packets)	Error Mean	Error Variance	Min Error	Max Error	Error Range
20	0.0011	0.0083	-0.3810	0.2508	0.6318
40	0.0014	0.0103	-0.3470	0.2703	0.6173
60	0.0018	0.0148	-0.3506	0.2952	0.6458
80	0.0021	0.0225	-0.4768	0.3312	0.8080
100	0.0020	0.0335	-0.6328	0.3887	1.0215
120	0.0017	0.0478	-0.7941	0.4878	1.2818
140	0.0016	0.0655	-0.9413	0.6200	1.5613
160	0.0025	0.0878	-1.0845	0.7476	1.8321
180	0.0033	0.1180	-1.1884	0.9116	2.1000
200	0.0029	0.1692	-1.3598	1.4350	2.7949

Table 5.2: Statistics of $\epsilon(re)$ for Stationary Traffic Measured from the Eth1/2 of Cisco1 in NCCT Network

distribution becomes wider with increasing of the buffer size, all of the errors are almost distributed in the area [-0.2 0.2] in the three curves, which indicates that the estimator is very effective to estimate the loss in the case of stationary traffic.

Non-Stationary Traffic

To check the applicability of the new estimator for variable traffic models, non stationary traffic is also generated to examine if the solution can handle such a case. Since the traffic is non-stationary, the proposed Kalman filter does not work on these conditions. Instead, the formula (5.2) and (5.3) will be used to calculate the packets mean and variance. In this experimental scenario, different types of DVD movies are played randomly. To

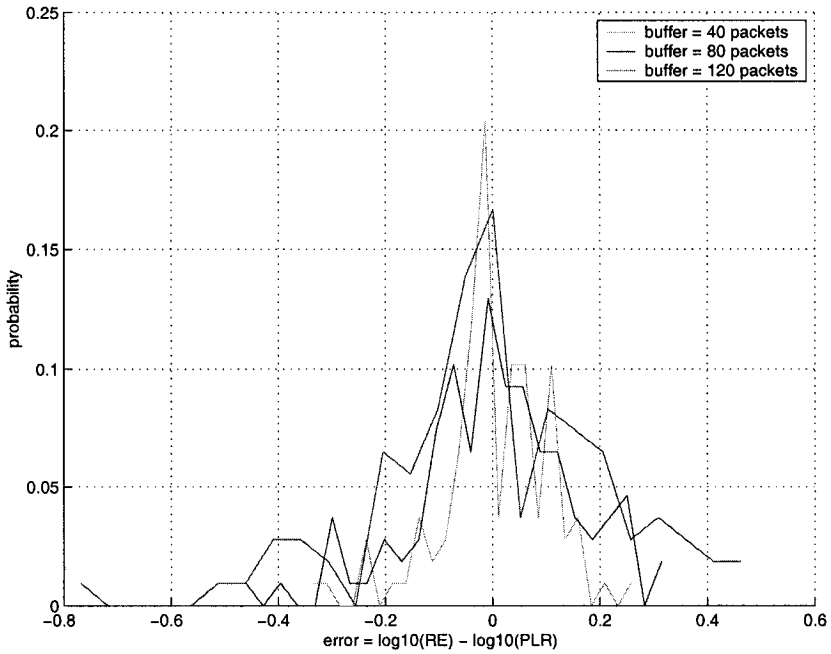


Figure 5.7: PDF of the Errors

increase the effect of variable DVD traffic on the aggregated input, the proportion of DVD flows is increased to 80 percent, and at the same time, the 40-flow background traffic is decreased to 20 percent. The utilization of the link remains 90 percents. Some samples of the traffic are measured and shown in Figure 5.8. That figure shows that both of the mean and variance of the traffic process vary with time. Therefore, in such a case the aggregated input traffic can be seen as non-stationary Gaussian traffic.

The window size p is still 3, and Figure 5.9 (I) and (II) show the measured loss ratio, *lbe* estimate and the reactive estimate as a sample path of the experiments with buffer size 60 and 120 packets, respectively. In the experiments, the loss probability is not stable, and the variation is high. However, the results also show that the new estimator still follows the actual loss ratio very closely after the initialization. Thus, even in a non stationary case, the new estimator has demonstrated its effectiveness.

The errors between the measured loss ratio and the estimate were also recorded as $\epsilon(re)$ for the variable buffer size from 20 packets to 200 packets, respectively. The values are shown in Table 5.3. The mean of the error from -0.111292 to 0.115372 is bigger than the one for the stationary traffic. However, it is still acceptable.

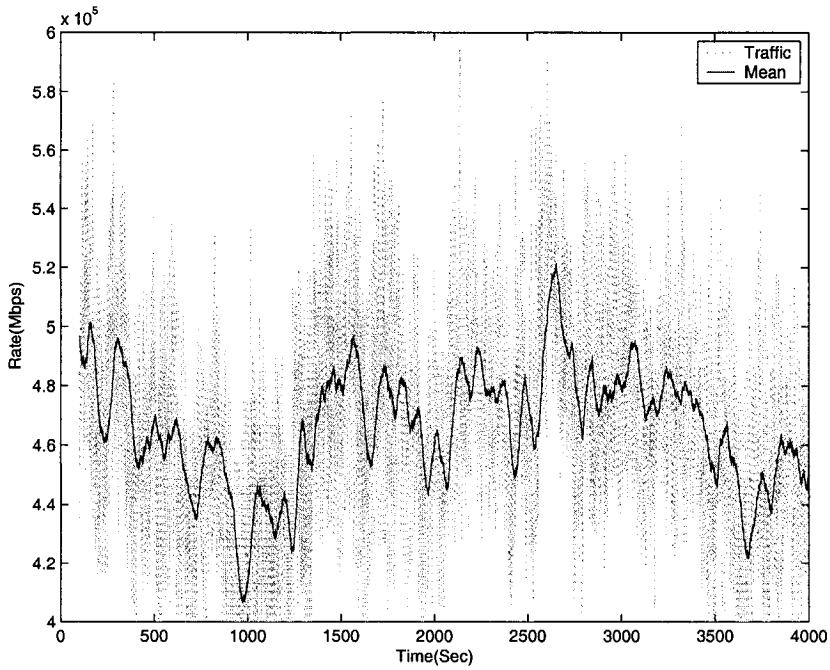
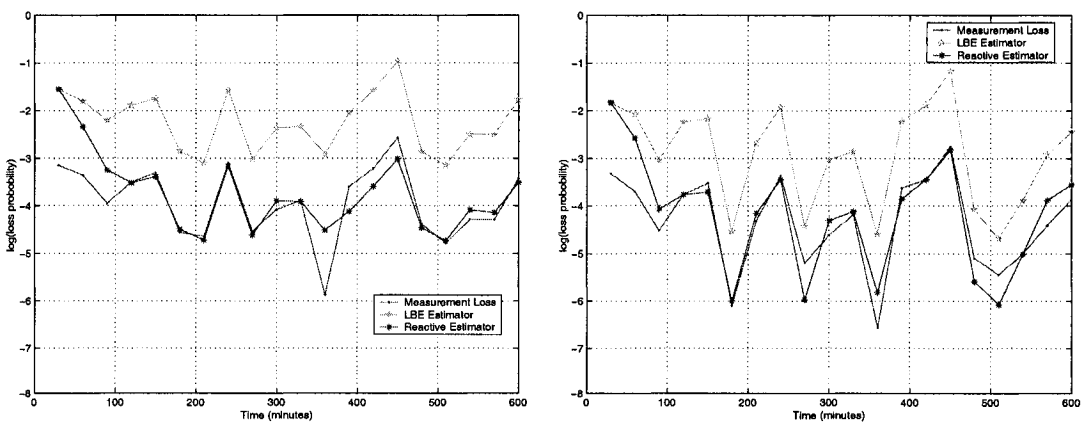


Figure 5.8: Traffic Measurement from the Eth1/2 of Cisco1 in NCCT Network



Buffer size = 60 packets **Buffer size = 120 packets**

Figure 5.9: Measured and Estimated Traffic Loss for Non-stationary Traffic Measured from the Eth1/2 of Cisco1 in NCCT Network.

Buffer (Packets)	Error Mean	Error Variance	Min Error	Max Error	Error Range
20	-0.034698	0.102226	-1.098979	0.955863	2.054842
40	-0.083795	0.102226	-1.100000	0.914420	2.014420
60	0.040773	0.103454	-1.036444	1.112889	2.149332
80	-0.138788	0.063094	-0.900000	1.114420	2.014420
100	-0.037986	0.067755	-0.874912	1.256548	2.131460
120	-0.111292	0.074771	-1.000000	1.014420	2.014420
140	0.037492	0.068867	-0.839356	1.378537	2.217894
160	0.038892	0.079840	-0.974357	1.230776	2.205133
180	0.114214	0.087819	-0.937965	1.410929	2.348894
200	0.115372	0.111716	-1.009184	1.234814	2.243998

Table 5.3: Statistics of $\epsilon(re)$ for Non-Stationary Traffic Measured from the Eth1/2 of Cisco1 in NCCT Network

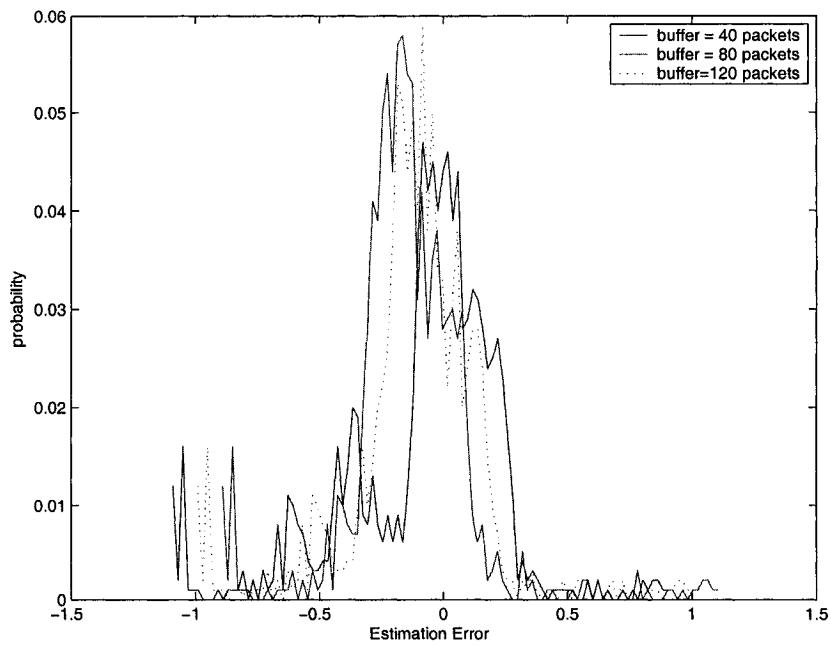


Figure 5.10: PDF of the Difference between the plr and the re.

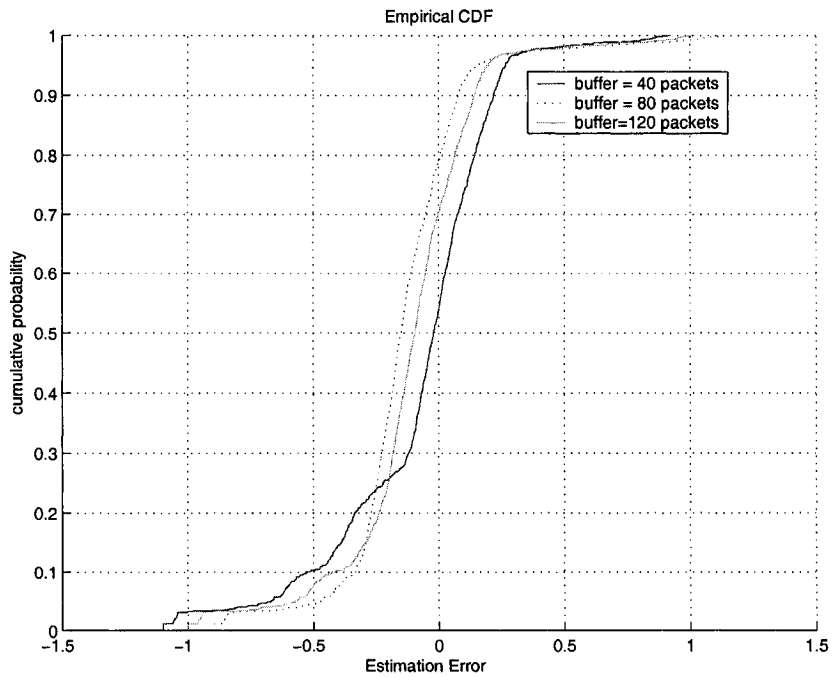


Figure 5.11: CDF of the Difference between the *plr* and the *re*.

In Figure 5.10 and Figure 5.1.4 error PDF and CDF are given, respectively. Compared with the stationary traffic, the non stationary traffic gives a more wide distribution for the errors. However, the errors are still almost distributed in the range [-0.5, 0.5] and this area is 80 percent of the total distribution. That means that the violation of QoS will be very little to happen. Consequently, the new estimator can still be used in the non stationary environment effectively.

5.2 Service Admission Control

The packet loss estimator will be used as the feedback transducer in the dynamic loss control system to maintain the QoS parameter within the guaranteed limits. But, first it will be applied to the service admission control and be evaluated whether or not the service admission control strategy can meet the QoS requirements alone.

5.2.1 Introduction

Based on the packet loss estimator, a practical Service Admission Control (SAC) scheme is proposed in this section. MPLS VPN services are provisioned as pipe model. In such a model, each VPN service is composed of several pipes, while each pipe is specified by the source PE, destination PE and the P nodes in its route.

The SAC strategy is run on a hybrid centralized and distributed base. The algorithm is composed by two parts: one is the control algorithm for the new VPN service in the centralized management workstation and the other one is the local algorithm on the distributed core P node.

When a new service request arrives in a PE node, the service request is firstly sent to the centralized service admission control server. Then the VPN path is derived according to the IP routing protocol, and the new request is sent to each p node on the path. For each core P node, there is a distributed admission control process, which measures its traffic and estimates the packet loss probability in a distributed way according. Finally all of the admission decisions made by the P nodes are sent back to the centralized service admission server, which makes the final decision.

5.2.2 Service Admission Control Algorithm

Each of the VPN service instances has a number of flows (i.e., pipes) that are routed through the provider network. Each service request contains a traffic specification for each of the node pairs that are part of the VPN. Each request is a duple that specifies the peak rates and average rates that are required. Suppose that there are M flows in the new VPN service. For the k^{th} flow, it is specified by the pair SD_k.

$$SD_k = (PE_{src},\ PE_{dst}),\ 1 < k < M$$

where, PE_{src} means the source PE node and PE_{dst} means the destination PE node.

It is also assumed that for each pair, the path is decided by the routing protocol such as RIP or OSPF in the service provider's network and all the intermediate nodes can be known in the path, denoted by set P_k

$$P_k = \{p_i \mid p_i \in path(k^{th})\}$$

where, $path(k^{th})$ means all of the P nodes in the path for the k^{th} flow.

Algorithm 1 SAC(New Service)

```
1:    for all pipe = 1 to M do
2:      P_m = GetAllNode( SD_m);
3:      for all node = 1 to NumberOf(P_m) do
4:        if (NSAC(pipe, node) == Rejected)
5:          Return Rejected;
6:        end if
7:      end for
8:    end for
9:    return Accepted;
```

Figure 5.12: Service Creation Algorithm

The service admission control algorithm on the centralized workstation follows the algorithm 1 shown in Figure 5.12.

The admission control algorithm will run in two loops for each new arrived VPN service, with the outer loop for the flow and the inner loop for each intermediate node in that path. The VPN service is accepted only when all the nodes of the path can accept the service. Otherwise it will be refused.

Figure 5.13 depicts the general service admission control process at each core P node. When a new service request is reached, the bandwidth decision is firstly made, where the bandwidth is examined to see if it can afford the new service request. If not, the new service request is refused. Otherwise, the packet loss probability is estimated based on the measured information of the aggregate traffic obtained from each interface of the P node and together with the user provided traffic description by applying the estimator equation. The estimated loss is compared with the preset QoS criterion, i.e. the preset loss value, and the error value is took for admission control system to decide whether or not the service is accepted. Generally speaking, if the new loss estimation is greater than the preset value, the service is rejected. On the other hand, the service is accepted.

5.2.3 Experimental Results

This section presents the experiment design and implementation issues. All of the measurements are collected from the NCCT & NCIT*net network [39]. The NCIT*net

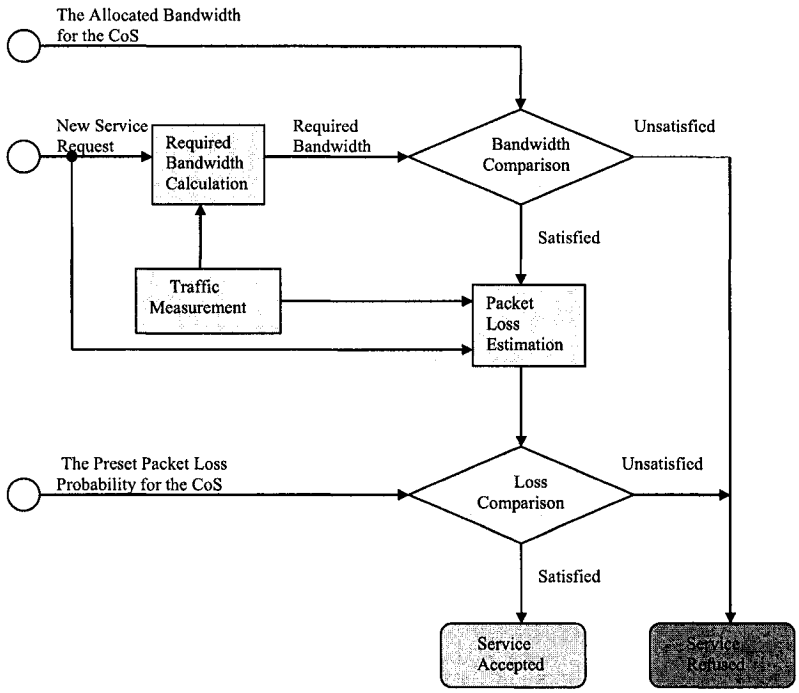


Figure 5.13: Service Creation Process on Each Node

can be considered as a live networking laboratory to support research in networking at the physical, networking and application layers. The NCCT lab is equipped with latest technologies used in internetworking, such as DWDM, SONET, ATM, Frame Relay, Ethernet, MPLS and IP. The topology of the whole network is shown in Figure 5.14.

Multiple VPN service flows are set between Cisco1 and Cisco2. The service activation and cancellation are configured on Cisco1 through the software system running on the management station. The block diagram of the service admission control software system is shown as in Figure 5.15.

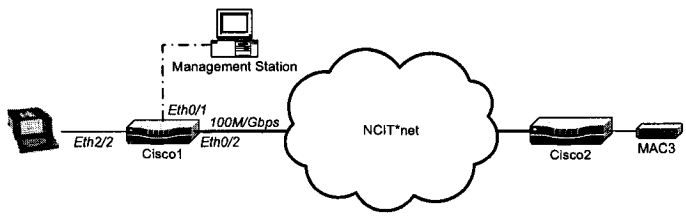


Figure 5.14: Block Diagram of the Testbed for the Service Admission Control System

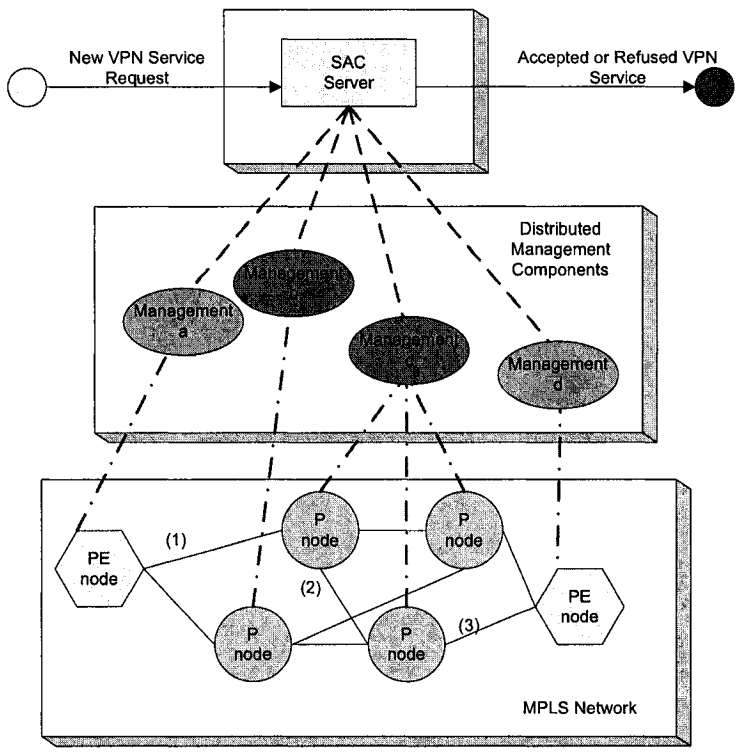


Figure 5.15: Block Diagram of the Service Admission Control System

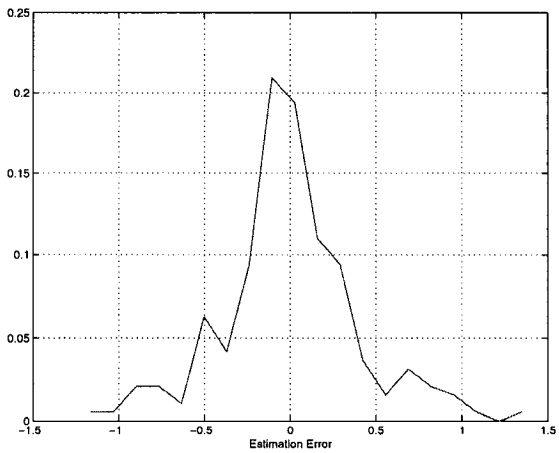


Figure 5.16: Violation of QoS Parameters on Stationary Traffic

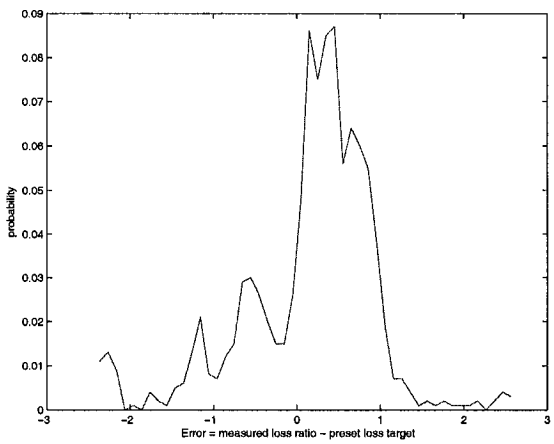


Figure 5.17: Violation of QoS Parameters on Non-stationary Traffic

QoS Violation

One of the evaluation criterions is the probability that the scheme results in the violation of the preset QoS. Lots of experiments are designed and tested on the live network. First, stationary traffic is tested. The following Figure 5.16 shows the PDF of the error caused by the controlling system. From the figure, one can know that the violation of QoS can be controlled within a small probability.

Then, non-stationary traffic is generated for each flow. Figure 5.17 depicts that the algorithm is not working pretty well, since the portion of QoS violation is large.

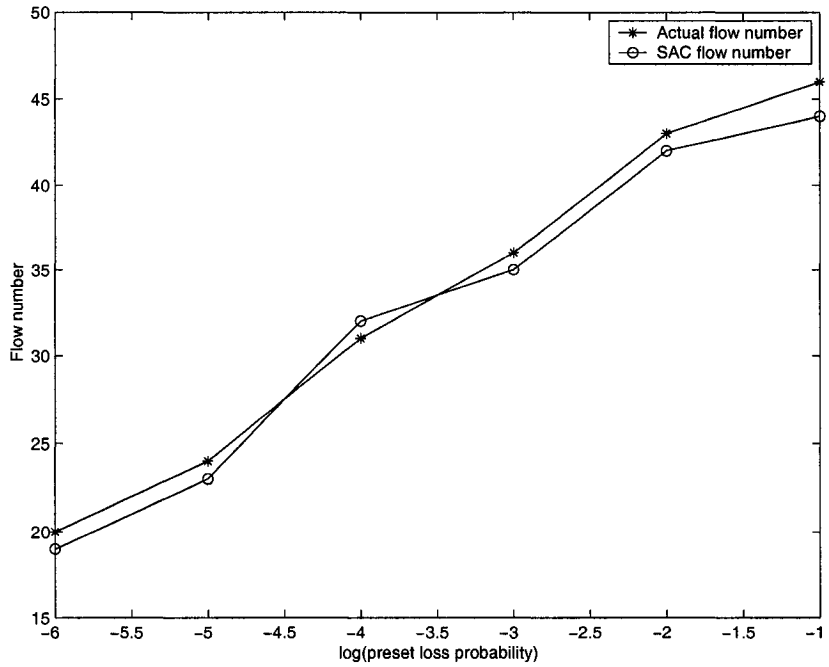


Figure 5.18: Comparison of Flow Numbers

Network Utilization

Another evaluation criterion is how high the level of network utilization an admission control algorithm can achieve while still meeting its service commitments. In the experiments, the GN Nettest is used to generate many same flows and it is tested how many flows can be accepted by the control system reference to a preset loss value. At the same time, the maximum flow number without violating the preset loss is also got by measuring the loss ratio. The Figure 5.18 compares the stationary flow numbers. The two numbers seem very close and that verifies the effectiveness of the admission control algorithm. However, when non-stationary flow traffic is aggregated, the flow numbers decrease sharply.

5.3 Dynamic Control of the Analytic Loss Model

In the last section, whe show that the measurement based service admission control system can give results that are quite far from the preset loss targets by its alone. Moreover the VPN service has a longer life span than the call service, so that admission

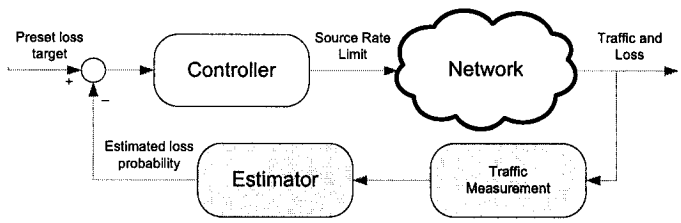


Figure 5.19: Online Loss Control Architecture

control is not applicable. This section will be given to maintaining QoS targets after the service has been provisioned.

Specifically, the packet loss probability control problem from a control theoretical point of view is approached. Since in most statistical QoS environments the packet loss probability is the premier QoS parameter, one considers to directly use the packet loss probability as the control signal. The signal has a slow dynamics compared with the buffer length. Through throttling the ingress traffic rate in the PE (Provider Edge) node, the packet loss probability in the core network is attempted to be maintained below the preset value.

The block diagram of control system is shown in Figure 5.19. To dynamically control congestion and maintain the packet loss in the provider's network, the traffic in the P node is monitored. Then, the measured traffic statistics are sent to the packet loss estimation process, which estimates the loss probability. The estimated loss probability and preset loss target are then fedback to the controller, where the maximum rate for each VPN service is calculated. These explicate rate signals are adjusted by the rate limit actuator process and then sent to the PE node through CLI commands. These commands are executed and take effect on the PE node to throttle the VPN input rates. All of these processes comprise the closed loop of the control system.

In the first preliminary approach, the dynamic loss model is analyzed and simplified. Then PI controller is designed for the model. Real traffic is measured and then simulated in Matlab to see whether the simplest approach can work.

5.3.1 Analytic Model of Packet Loss System

In the loss control system, since each P node is considered to implement finite buffer FIFO scheduling, the state variable is defined to be the momentary length of the queue for each output link i at time k, denoted by $q_i(k)$. Let $\alpha_i(n)$ denote the input process to link i. That is, $\alpha_i(k)$ equals the number of packets that arrive during time slot $[k, k+1)$, $k = 0, 1...n$. Then the state evolution of buffer i is described by the following difference equation:

$$q_i(k+1) = min([q_i(k) + \alpha_i(k) - c]^+,\ b) \tag{5.30}$$

where $[x]^+ = max(x, 0)$, b is the maximal buffer size allocated for link i and c is the bandwidth of link i.

Let $r_{ab}(k)$ denote the rate at which VPN connection (ab) has been admitted to the network during $[k, k+1)$ and link i is one part of the connection from PE node a to another PE node b. Then the total amount of traffic arriving at buffer i during [k, k+1) is:

$$\alpha_i(k) = \sum [r_{ab}(k - \delta_{ai})] \tag{5.31}$$

$$\delta_{ai} = \tau_{ai}/T \tag{5.32}$$

where, τ_{ai} is the time delay along the path from the source PE node a to P node i, which includes transmission, propagation and processing delays. δ_{ai} means the delay measured in time slots T. In order to make the system simple, δ_{ai} is thought as an integer in the paper. From (5.30) and (5.31), the following state equation can be derived:

$$q_i(k+1) = min([q_i(k) + (r_{ab}(k - \delta_{ai})) - c]^+, b) \tag{5.33}$$

Let $l_i(k)$ denotes the number of packets lost in time period $[k, k+1)$ from the link i. Then it can be calculated as follows:

$$\begin{aligned} l_i(k) &= [q_i(k) + \alpha_i(k) - c]^+ - b]^+ \\ &= [q_i(k) + \alpha_i(k) - c - b]^+ \end{aligned} \tag{5.34}$$

Since only the case of a single congested node (link i) is considered in the following design section, we do not write the notation i in the equations. Therefore, the system state equations are expressed as follows:

$$q(k+1) = min([q(k) + (r_{ab}(k - \delta_a)) - c]^+, b) \tag{5.35}$$

$$l(k) = [q(k) + (r_{ab}(k - \delta_a)) - c - b]^+ \tag{5.36}$$

5.3.2 Controller Design

The controller is composed with two parts: the control algorithm and the rate admission algorithm. The control algorithm calculates the admission rates $R_{ab}(k)$ for each source based on the estimated packet loss probability, while the rate admission algorithm determines how the PE node reacts to these admission rates and generates the admitted rates $r_{ab}(k)$.

First, in the rate admission algorithm, the source PE selects the admitted rate as the smallest among all admission rates corresponding to the link and the desired transmission rate. That is expressed in the follows:

$$r_{ab}(k) = min((R_{ab}(k + 1 - \delta_{ia})),\ r^0_{ab}(k)) \tag{5.37}$$

where $r^0_{ab}(k)$ denotes the offered traffic load by CE nodes. δ_{ia} is the feedback delay of connection (ab) with respect to link i, which is defined in a similar way as the forward delay. Since the feedback path is the reverse of the forward path, the two delays are assumed to be equal in the paper and are denoted as half of the total round delay (δ). The following equations make the relations clearly.

$$\delta_{ai} = \delta_{ia} = \delta_a \tag{5.38}$$

$$\delta = 2\delta_a \tag{5.39}$$

Second, the admission rate is calculated in the control algorithm by applying the classic PID design. The PID controller is a standard compensator that is widely used in many applications. Its basic properties can be found in [47]. Due to the slow output signal dynamics, the derivative component of the controller can be set to zero. Therefore, the study only proposes a PI compensator. A PI compensator, denoted by $D(z)$, consists of two terms: proportional and integral component. It has the form as:

$$D(z) = K_p + K_i \frac{z}{z - 1} \tag{5.40}$$

5.3.3 Stability Considerations

In the section, the stability analysis for the linear system is emphasized. In control theory, stability means that for any bounded input over any amount of time, the output will

also be bounded. This is known as Bounded-input, bounded-output (BIBO) stability. If a system is BIBO stable then the output cannot blow up if the input remains finite. Mathematically, this means that for a linear continuous-time system to be stable all of the poles of its transfer function must lie in the closed left half of the complex plane if the Laplace transform is used (i.e. its real part is less than or equal to zero) or lie on or inside the unit circle if the z-transform is used (i.e. its modulus is less than or equal to one)

In packet loss system, after making (5.35) and (5.36) linear and applying the z-transform to the plant state equation, we can get the following z-transform function for the plant.

$$\begin{aligned} L(z) &= Q(z) + R(z)z^{-\delta_a} \\ &= \frac{z}{z-1}R(z)z^{-\delta_a} \end{aligned} \tag{5.41}$$

In order to study the asymptotic stability properties, the dynamic equations of the system (5.35) and (5.36) are simplified by removing the saturation nonlinearities. After making (5.35) and (5.36) linear and applying the z-transform, the following z-transform functions for the plant are derived:

$$Q(z)z = Q(z) + R(z)z^{-\delta_a} \tag{5.42}$$

$$L(z) = Q(z) + R(z)z^{-\delta_a} \tag{5.43}$$

where $L(z)$ is the z-transform of $l(k)$, $Q(z)$ is the z-transform of $q(k)$ and $R(z)$ is the z-transform of $r_{ab}(k)$.

After combining the previous two equations, the closed-loop transfer equation of the plant, $G(z)$, is written as:

$$G(Z) = \frac{L(z)}{R(z)} \tag{5.44}$$

$$= \frac{Q(z)}{R(z)} + z^{-\delta_a} \tag{5.45}$$

$$= \frac{z}{z-1}z^{-\delta_a} \tag{5.46}$$

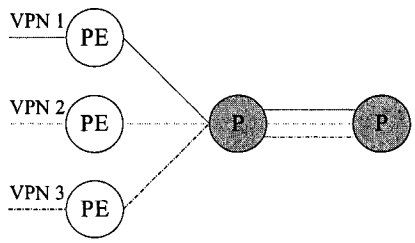


Figure 5.20: Topology of the Simulated Network

When the digital controller and feedback delay are applied to the system, the closed-loop equations between the output $L(z)$ and the target input $L_0(z)$ is expressed as:

$$\frac{L(z)}{L_0(z)} = \frac{D(z)G(z)}{1 + D(z)G(z)z^{-\delta_a}} \tag{5.47}$$

Therefore, the characteristic equation is

$$1 + D(z)G(z)z^{-\delta_a} = 0 \tag{5.48}$$

Apply (5.40) into (5.48), then the characteristic equation is changed to:

$$1 + (K_p + K_i\frac{z}{z-1})(\frac{z}{z-1}z^{-\delta_a})(z^{-\delta_a}) = 0 \tag{5.49}$$

which can be simplified to the following equation:

$$z^{\delta+1} - 2z^{\delta} + z^{\delta-1} + (K_p + K_i)z - K_p = 0 \tag{5.50}$$

To make the system stable, all the roots of (5.50) must lie inside the unit circle in the complex z-plane according to the stability definition. Two parameters Kp and Ki is obtained by the well-known root locus method or Nyquist diagram.

5.3.4 Experimental Results

The performance of the control system is evaluated in NCCT lab. For this purpose, the configuration diagram shown in Figure 5.20 is used that consists of three VPN traffic classes each fed by MPEG2 UDP sources and Ethernet traffic. Link P-P is where service aggregation happened. In the experiment, all the link capacities are 10 Mbps. The basic time interval is selected as 1s. The delay δ is fixed at 10. The loss ratio is measured through averaging 5 time periods, i.e., the moving window size is 5s. The window size for traffic mean and variance is also selected as 5s.

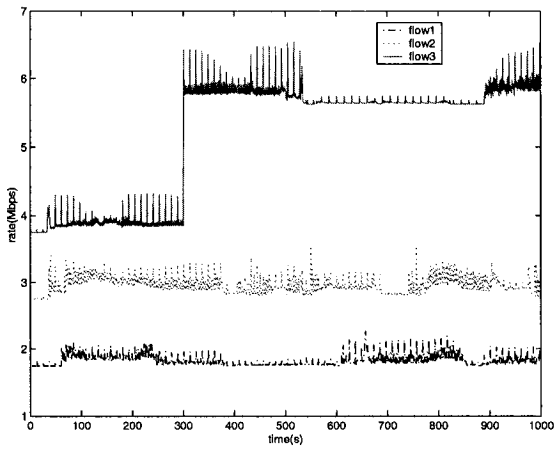


Figure 5.21: Input Packet Rates of Three VPN Traffic

The input traffic of the three VPNs are shown in Figure 5.21. The average rate of VPN 1 is about 4 Mbps in time window [0, 300s] and is increased to 6 Mps after time 300s. The mean value of VPN 2 is about 3 Mbps and VPN 3 is 2 Mbps. The loss target is preset to 10%. Therefore, before time 300s, the loss in the network is below this target without activating the control algorithm.

Transient Performance Analysis

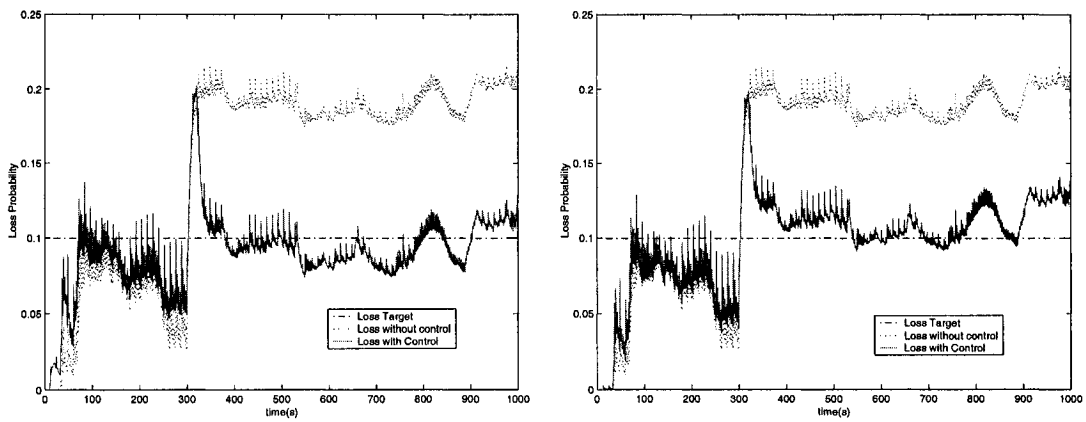
The control is activated by controlling the traffic rate of the output interface at the PE nodes. The CE node injects non-elastic traffic based on the traces generated by MPEG 2 movies and Ethernet traces. A value for the QoS loss target is chosen and the controller has to throttle the traffic to ensure the target is not breached.

Equation (5.50) has two parameters that have to be calculated: K_p and K_i. They can be obtained by the well-know root locus method or Nyquist diagram [47]. Two sets of parameter K_p and K_i, detailed in Table 5.4 5.3.4, are used to see how the system behavior varies. The value for buffer is fixed at 20ms.

	K_p	K_i
1	0.3	0.08
2	0.2	0.04

Table 5.4: K_p and K_i

To verify the effectiveness of the control system, firstly the loss ratio is measured



(I) First set of PI (II) Second set of PI

Figure 5.22: Loss Probability Time Series when Controlled by the PID Controller

without the control system. The result is shown in Figure 5.22 (I) and (II) as the *loss without control.* It is obviously that the loss ratio is increased to a big value (20%) when there is congestion after time 300s.

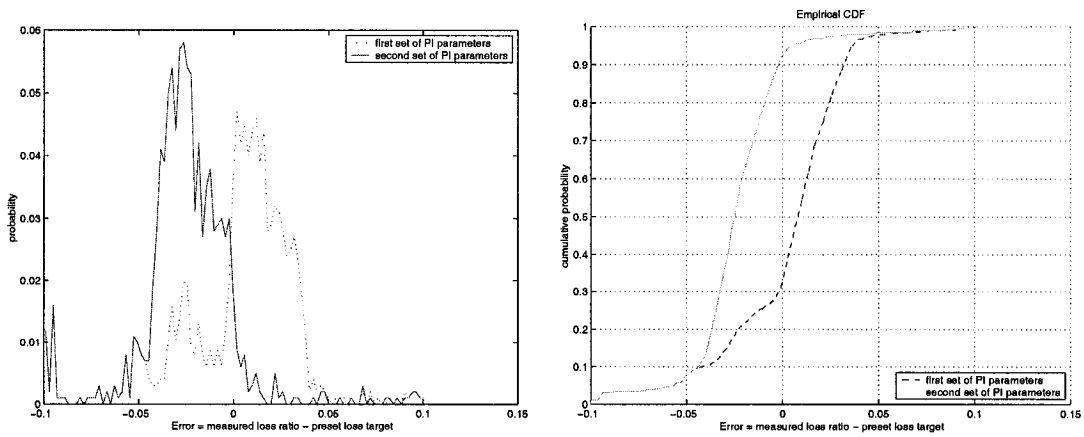
When the first set of control parameters is selected, the loss curve is shown in Figure 5.22 (I) as *loss with control.* It is seen that after congestion happened, the loss (20%) is controlled to be down back to the preset value (10%). The response time is about 50s. In such a case, there is no oscillatory behavior.

When the second set of control parameters in Table 5.4 is selected, the performance is shown in Figure 5.22 (II) as *loss with control.* This time, the loss target is also achieved. But the response time is increased to 250s. When the K_p is selected as bigger than 0.4, the system is showing an oscillatory behavior. And when the K_p is equal to 0.1, the system needs a very long time to reach the preset target. In addition, the K_p is smaller than 0.1, the system cannot reach the preset value in the simulation time period.

Steady State Error

An important characteristic of a control system is its ability to follow, or track, certain inputs with a minimum of error. In this section the effects of the system on the steady-state system errors are considered.

For the system described by (5.47), the system error $e(z)$ is defined as the difference



pdf cdf

Figure 5.23: Distribution of the Error between the Measured Loss Ratio and Preset Loss Target

between the system input and the system out. That is:

$$E(z) = L(z) - L_0(z) \tag{5.51}$$

It is simplified as the followings:

$$E(z) = L_0(z) - \frac{D(z)G(z)}{1 + D(z)G(z)z^{-\delta_a}} L_0(z) \tag{5.52}$$

$$= \frac{1 + D(z)G(z)z^{-\delta_a} - D(z)G(z)}{1 + D(z)G(z)z^{-\delta_a}} L_0(z) \tag{5.53}$$

The steady-state error will be derived for the common input: a step input. For the unit-step input,

$$L_0(z) = \frac{z}{z-1}$$

Then, from the final-value theorem, the steady-state error is seen to be

$$e(kT) = \lim_{z \to 1}(z-1)E(z) \tag{5.54}$$

If there is a limited value for the equation (5.54), the steady-state error is obtained.

To verify the effectiveness of maintaining the loss target in steady state, a number of experiments are designed and the errors between the measured loss ratio and the preset loss target are recorded. These experiments are designed through changing the preset loss target, the input rates and buffer size of the P node.

In Figure 5.23 (I), the analysis of the probability density functions shows that in all cases the control system keeps the actual loss close to the target loss. The error of -0.1 represents the period when there are almost no loss because of unloaded traffic. Figure 5.23 (II) shows the cumulative probability density function, from which one can see that there is a control error but that the expectation of the error is very close to zero. About 90% of the errors are in the region [-0.05, 0.05], which shows the effectiveness of the control system.

5.4 Conclusion

In the chapter, a new packet loss estimator is proposed to extend the previous estimators used for building transducers for packet loss control loops in MPLS VPN networks. Experiments for the evaluation of the quality of the transducers has been designed and preformed. The results show that the proposed new estimator can calculate the loss probability more accurately. Then, the proposed service admission control method measures the aggregate traffic effective bandwidth and loss ratio, estimates the packet loss probability, and derives the admission control decision. The numerical evaluation result shows that in stationary traffic case, the algorithm performs very effectively. However, the control scheme is not good for the non-stationary traffic because of the unpredictable traffic characteristics. The closed-loop control mechanism is proposed to dynamically control the QoS for the existed services. However, the simplified model should be improved to meet the QoS requirements. In the next chapter, the dynamic model will be identified with the system identification method.

Chapter 6

System Identification of the Packet Loss Model

In the proposed loss control system, modeling of the dynamics of the plant is one of the key issues to be addressed. In the previous chapter, the analytical mathematical model is constructed based on the mechanisms that govern the system's behavior as queues stored in a buffer which is emptied by one of scheduling policies such as FIFO and GP. A PI controller is then designed to control the model. However, the performance shows that controlling of the directly linearized model results in a large error. The reason may be because the knowledge of the system's mechanism is incomplete; or the constructed model is simplified too much to present the real system well enough. This chapter tries to identify the packet loss system by applying the system identification methods, where the ingress packet rates are measured as input signals and the packet loss probability is estimated as the output signal. Both of the offline and online Prediction Error Methods (PEM) are investigated. Lots of experiments are designed on the live NCIT*net2 network to evaluate and verify the effectiveness of the algorithms.

6.1 Introduction to System Identification

The packet loss system is a *dynamic* system, since its current output value depends on not only the current external inputs but also on their earlier values. To analyze and control such a dynamic system, mathematical models are usually employed to describe the relationship among certain variables of the system. There are basically two approaches to the problem of building a mathematical model of a given system [76].

1. The first procedure is called direct modeling. Based on the physical laws and

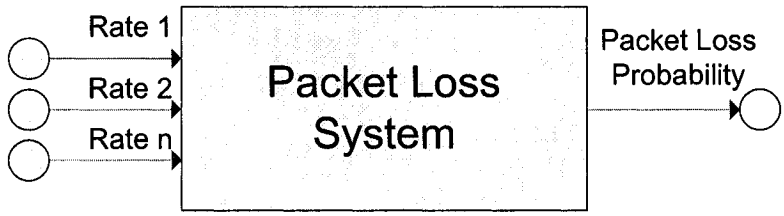


Figure 6.1: Packet Loss System Model

relationships that govern the system's behavior, a mathematical model can be obtained. In this procedure, it is required to look into the mechanisms that generate signals and variables inside the system.

2. Sometimes, it is impossible for direct modeling; for example, the system may have incomplete mechanisms or unpredictable properties. Moreover, some derived models are unnecessarily complex. In such cases, an experimental approach is used to construct the model based on signal measurements from the system. The approach is known as the *System Identification* method.

In practice, system identification method usually builds mathematical models of dynamic systems based on both prior system knowledge and observed experimental data, and is thus a combined theoretical and experimental modeling method.

The procedure of system identification has a natural logical flow: based on prior knowledge, experiments are designed. The required data are then sampled and obtained from the experiments. A preferred model is selected from the model set and some fit criterion is made. The parameters of the preferred model are identified with the selected fit criterion based on the measured data. The method of least squares is the most well-known fit criterion, which fits the calculated values to the measured outputs by the least square fit value. If the identified model does not pass the model validation test, it must go back and revise the various steps of the procedure until some satisfied model is identified.

Considering the packet loss system as shown in Figure 6.1, where the traffic rates are inputs and packet loss probability is the output, a mathematical model is sought to be obtained via plant identification techniques as developed for control systems. The

model will be identified using system identification method as developed in this chapter. In the following sections, the linear state model is first selected and initial parameters are obtained. PEM algorithm is then designed to identify the model parameters. The experiments are proposed and the sampling frequency is decided. The numeric results are evaluated to verify the effectiveness of the algorithm. Finally, the Recursive Parameter Estimation algorithm (RPE) is applied to deal with the time-varying system behavior.

6.2 Model Identification Method

Linear modeling is the most common way to describe a dynamic system. The linear model identification and control techniques are more mature than their nonlinear counterparts. Moreover, linear models are relatively easier to apply and implement. For most of non-linear cases, they are translated into linear models by appropriate linearization methods. Since the model is just a description of major concerned properties of the system, the model does not need to be a true and accurate description of the system, nor should one believe it to be, as long as it serves its purpose. In this study, the linear state space model is assumed for the dynamic packet loss system.

6.2.1 Initial Model Selection

Based on the mechanisms that govern the packet loss system's behavior, an initial mathematical model is obtained. The packet loss system is assumed as a linear time-invariant model at the beginning of the model analysis. Since each core network node is considered to implement the finite buffer FIFO scheduling, the state variable is defined to be the momentary length of the queue for each output link i at time k, denoted by $q_i(k)$. Let $\alpha_i(k)$ equal the number of packets that arrive during time slot $[k, k+1)$, $k = 0, 1...n$. Therefore, the state evolution of buffer i is described by the following difference equation:

$$q_i(k+1) = min([q_i(k) + \alpha_i(k) - c]^+,\ b) \tag{6.1}$$

where $[x]^+ = max(x, 0)$, b is the maximal buffer size allocated for the link i and c is the bandwidth of the link i.

Let $r_{ab}(k)$ denote the packets at which VPN connection (ab) has been admitted to the network during $[k, k+1)$ and the link i is one part of the connection from the edge node a to another edge node b. Then the total amount of traffic arriving at buffer i during [k, k+1) is:

$$\alpha_i(k) = \sum r_{ab}(k - \delta_{ai}) \tag{6.2}$$

$$\delta_{ai} = \tau_{ai}/T \tag{6.3}$$

where τ_{ai} is the time delay along the path from the source edge node a to the core node i, which includes transmission, propagation and processing delays. δ_{ai} means the delay measured in time slots T. In order to make the system simple, δ_{ai} is thought of as an integer in the thesis. From (5.30) and (5.31), the following state equation can be derived:

$$q_i(k+1) = min([q_i(k) + \sum r_{ab}(k - \delta_{ai}) - c]^+, b) \tag{6.4}$$

Let $l_i(k)$ denote the number of packets lost in time period $[k, k+1)$ from the link i. Then it can be calculated as following:

$$l_i(k) = [[q_i(k) + \alpha_i(k) - c]^+ - b]^+ \tag{6.5}$$

$$= [q_i(k) + \alpha_i(k) - c - b]^+ \tag{6.6}$$

Since only the case of one single congested node $(i = 1)$ is considered, the notation i can be neglected in the equations. The system state equations are expressed as following:

$$q(k+1) = min([q(k) + \sum r_{ab}(k - \delta_a) - c]^+, b) \tag{6.7}$$

$$l(k) = [q(k) + \sum (r_{ab}(k - \delta_a) - c - b]^+ \tag{6.8}$$

After making (6.7)and (6.8) linear and neglecting the time delay, the two equations are changed to:

$$q(k+1) = q(k) + r(k) \tag{6.9}$$

$$l(k) = q(k) + r(k) \tag{6.10}$$

Applying the z-transform to the plant state equations, the following z transform

function for the plant is derived.

$$l(z) = q(z) + r(z) \tag{6.11}$$

$$= \frac{1}{z-1} r(z) + r(z) \tag{6.12}$$

$$= \frac{z}{z-1} r(z) \tag{6.13}$$

Since in most statistical QoS environments the packet loss probability is the premier QoS parameter, the packet loss probability is directly looked at as the output signal.

The packet loss probability in k^{th} interval, denoted by $L(k)$, can be simulated by PLR and simply calculated as:

$$L(k) = \frac{\sum_{i=1}^{k} l(i)}{k * C * \rho} \tag{6.14}$$

where C is the bandwidth of the interface and ρ is the long-term utilization of the link. Both of them can be assumed as constant values. After z-transform, it gives:

$$L(z) = \frac{z}{z-1} l(z) \tag{6.15}$$

Therefore, the following simplified transfer equation is derived:

$$\frac{L(z)}{r(z)} = \frac{z^2}{z^2 - 2z + 1} \tag{6.16}$$

A linear discrete time-invariant model is usually specified by the impulse response, the spectrum response and the transfer function. However, the state-space model is a more common representation of the dynamic model, where it is written as a system of first-order difference equations using an auxiliary state vector $x(k)$. Since within the transfer function the order is 2, the state-space model of the packet loss system is then simplified as a 2-order IIR system:

$$x(k+1) = Ax(k) + Br(k) \tag{6.17}$$

$$L(k) = Cx(k) + Dr(k) + e(k) \tag{6.18}$$

where $x(k)$ is the vector of states, $L(k)$ is the output, $r(k)$ is the input and $e(k)$ is the measurement noise in the k^{th} time interval.

To identify the state-space model of the dynamic system, most of the effort involved relates to defining and manipulating the structure. That is how to determine the structure and values of matrix A, B, C and D.

The effect of the input on the output will be delayed at least one sample, so D is assumed to be zero in order for that there is no direct influence from $r(k)$ to $L(k)$.

For a given system, there is no unique state-variable formulation. The canonical form of the state-variable model is selected in the thesis to simplify the processing. The initial model structure is described as the following four matrixes in the canonical form:

$$A = \begin{bmatrix} 0 & 1 \\ \alpha_0 & \alpha_1 \end{bmatrix} B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} \beta_0 & \beta_1 \end{bmatrix} D = \begin{bmatrix} 0 & 0 \end{bmatrix} \tag{6.19}$$

where the system transfer function $G(z)$ is then seen as:

$$G(z) = \frac{L(z)}{r(z)} = C[zI - A]^{-1}B + D \tag{6.20}$$

There are four parameters, α_0, α_1, β_0 and β_1 should be identified in the experiments.

6.2.2 Parameter Estimation

After the model structure is selected, the set of candidate models $M(\theta)$ is defined using a parameter vector θ denoted with the following equation.

$$\theta = (\alpha_0, \alpha_1, \beta_0, \beta_1)^T \tag{6.21}$$

where $\alpha_0, \alpha_1, \beta_0, \beta_1$ are as in (6.2.1).

The set of model is defined as:

$$M^* = \{M(\theta) | \theta \in D_m\} \tag{6.22}$$

The search for the best model within the set then becomes a problem of determining or estimating the proper parameter θ^*. The proper parameter θ^* should be selected in order to satisfy that $M(\theta^*)$ is the best model in all of the model set M^*.

Since the essence of a model is its prediction aspect, the model performance is judged in this respect. Prediction Error Method (PEM) is considered as a kind of generalized

framework for system identification. With PEM, model parameters will be estimated by minimizing the optimally determined one-step-ahead output prediction error.

Given the particular parameters θ^* and the measured data set Z,

$$Z = \{z_k | z_k = [L(k), r(k)]\}, k \in [1..N]$$

the best one-step ahead output prediction at k is:

$$\hat{L}(k|\theta^*) = (C[I - A]^{-1}B + D)r(k-1) \tag{6.23}$$

Then, the one-step ahead output prediction error at k, denoted by $\epsilon(k, \theta^*)$, is given by

$$\epsilon(k, \theta^*) = L(k) - \hat{L}(k|\theta^*) \tag{6.24}$$

The PEM can be seen to minimize the prediction errors defined by a scalar-valued function $f(.)$. That is, the estimated $\hat{\theta}$ is defined by minimization of the following norm:

$$\hat{\theta} = argmin_\theta V(\theta) \tag{6.25}$$

$$V(\theta) = \frac{1}{N}\sum_{k=1}^{N} f(\epsilon(k, \theta)) \tag{6.26}$$

For the choice of $f(.)$, this thesis uses a quadratic norm, which is usually a standard choice in most studies:

$$f(.) = \frac{1}{2}\epsilon^2 \tag{6.27}$$

Then the parameter estimation is to search the θ^* to minimize the following value $V(\theta)$:

$$V(\theta) = \frac{1}{2N}\sum_{k=1}^{N}(\epsilon(k, \theta))^2 \tag{6.28}$$

In general, (6.28) cannot be minimized by analytical methods. The solution has to be found by iterative, numerical techniques. From the initial estimate, a Gauss-Newton minimization procedure [75] is carried out until the norm of the Gauss-Newton direction is less than a certain tolerance. With Gauss-Newton algorithm, the iterative searching routine updates the estimate of the minimizing point according to

$$\hat{\theta}^{i+1} = \hat{\theta}^i - [H(\hat{\theta}^i)]^{-1}V'(\theta^i) \tag{6.29}$$

where $\hat{\theta}^i$ denotes the i^{th} iterate for the estimate of θ and $V'(\theta)$ denotes the gradient of $V(\theta)$, and $H(\theta)$ is the Hessian matrix that modifies the search direction. The selection of $H(\theta)$ is obtained using the numerical procedure of the descent algorithm and guarantees convergence to a stationary point. It is calculated by the following equation:

$$H(\theta) = \frac{1}{N}\sum_{k=1}^{N}\psi(k,\theta)\psi^T(k,\theta) \tag{6.30}$$

where $\psi(k,\theta)$ denotes the gradient of the error. That is,

$$\psi(k,\theta) = \epsilon'(k,\theta)$$

The major burden lies in the calculation of the sequence of prediction errors and their gradients. The calculation of $H(\theta)$ and of the θ is done by using *pem* in this thesis.

6.2.3 Model Order Validation

In the previous section, the order of the model is derived from the equations. This experiment studies whether or not other higher-order models give better prediction results than the selected low-order model.

Of the formal order-estimation methods, perhaps the most well known are the Akaike Information Criterion (AIC) and the Final Prediction Error (FPE), both introduced by Akaike [1]. They simulate the cross validation situation, where the model is tested on another data set.

The FPE is calculated as

$$FPE = \frac{1+n/N}{1-n/N} * \mathcal{V}$$

where n is the total number of estimated parameters and N is the length of data record. $\mathcal{V}$ is the loss function or quadratic fit for the structure in question and is defined as:

$$\begin{aligned}\mathcal{V} &= \sum_{k=1}^{N}\epsilon^2(k) \\ &= \sum_{k=1}^{N}(L(k)-\hat{L}(k))^2\end{aligned} \tag{6.31}$$

where $L(k)$ is the measured packet loss probability and $\hat{L}(k)$ is the estimated packet loss probability according to the model.

In a similar way, the AIC is formed as

$$AIC \approx log[(1 + 2n/N) * \mathcal{V}]$$

According to Akaike's theory, in a collection of different models, which will best fit the plant model accurately for a given error is the one with the smallest FPE or AIC.

6.3 Data Collection

The System identification problem is to estimate a model of a system based on observed input-output data. Several ways exist to describe a system and to estimate such descriptions. This section presents issues to design network experiments and to prepare the input-output data from measurement technologies. All of the experiments are designed across the National Capital Institute of Telecommunications network (NCIT*net2).

The testbed is setup as the same as the one used in Chapter 4. shown in Figure 4.3, where three BGP/MPLS VPN services are configured across the NCIT*net2 network. One management workstation is connected with all of the devices through Interface eth0/1 in the Cisco1 to measure the VPN traffic.

6.3.1 Data Collection

The total traffic in each time interval k is measured as sum of packets of each VPN flow. It is expressed as:

$$\hat{A}(k) = \sum_{i=1}^{M} \hat{A}_i(k) * S \tag{6.32}$$

where $\hat{A}(k)$ is the aggregate traffic in the k^{th} time interval; and $\hat{A}_i(k)$ is the received number of output packets from VPN source i in the time interval. In this testbed, M = 3. S is the sample rate configured for the sFlow, which means the sFlow agent will analyze one sample packet for every S packets.

If the time interval is T, then the input packet rates $r(k)$ for the k^{th} time interval can be calculated according to (6.33):

$$r(k) = \frac{\hat{A}(k)}{T} \tag{6.33}$$

Packet loss probability is the output of the model. It is estimated with a recursive linear estimation algorithm proposed in Chapter 5. Section 1. The estimator is shown in

(5.11), which is composed on of two parts: the Large Buffer Estimator (LBE), described as:

$$LBE = -\frac{2b(c-\hat{\mu})}{\hat{\sigma}^2}log(e) - log(\frac{2\hat{\mu}(1-\hat{\mu})}{\hat{\sigma}^2}) \tag{6.34}$$

where b is the buffer size of the router, c is the bandwidth of the output, $\hat{\mu}$ is the mean of the measured traffic and $\hat{\sigma}^2$ is the variance of the measured traffic.

The other part is the measured Packet Loss Ratio (PLR). In the experiments, to measure PLR of the aggregate traffic in the core link, interface counters in MIB II are also exported through SNMP protocol. The PLR can be calculated by the following equation:

$$PLR = \frac{N_l}{N_l + N_O} \tag{6.35}$$

where N_l means the number of the total lost packets and N_O is the number of the transported packets. Both are retrieved from the interface counters in standard MIB II from the Cisco1 router.

Based on the PLR and LBE, the packet loss probability $L(k)$ is estimated for each time interval k. The details should be found in Chapter 5.

Then the original data set for the system identification is ready and denoted in the following equation.

$$Z = \{z_k | z_k = [L(k), r(k)]\}, k \in [1..N] \tag{6.36}$$

6.3.2 Determining the Sampling Period

The procedure of sampling data is inherent in computer based data acquisition systems. It is unavoidable that sampling leads to information losses. In order to obtain full significant information from on-line measured data, it is necessary to determine the sampling frequency so that the full spectral information is not lost from the process.

The issue can be resolved by sampling the process and then finding its Power Spectral Density (PSD) via the periodogram (sample spectrum)[57]. The periodogram is estimated by calculating the discrete-time Fourier transform of the autocorrelation function of the signal. If X_n is a stationary stochastic process, the PSD of this signal is defined in the following equation.

$$PSD(\omega) = \frac{1}{2\pi} \sum_{m=-\infty}^{\infty} R_{xx}(m) e^{-j\omega m} \tag{6.37}$$

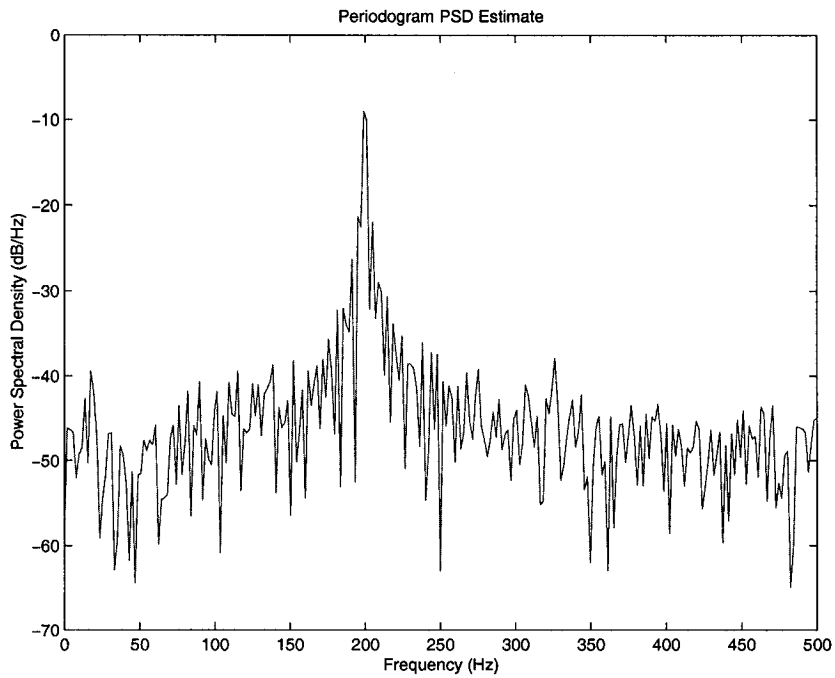


Figure 6.2: PSD of the Measured Packet Loss Data

where $R_{xx}(m)$ is the autocorrelation and defined by the equation (6.38).

$$R_{xx}(m) = E\{X_{n+m}X_n\} \tag{6.38}$$

Then the cut-off frequency is found from the PSD, and by using Shannon's sampling theorem [77], a minimum sampling frequency is determined.

First, the sampling interval is set to 0.25 seconds and there are 4000 data points collected. The periodogram of the process is taken and shown in Figure 6.2. From the cut-off frequency it is determined that sampling should be performed at a frequency of one sample every two seconds or better to obtain significant spectral information for the process. Therefore, in the next experiments T is set to two seconds.

6.3.3 Data Preprocessing

For the off-line model identification, five data samples are recorded with different types of video traffic. Each sample is composed of 500 points which is captured from a 20-minute real-time traffic. Since the system is treated as a linear system in the following identification part, the direct components of the input and output signals are removed.

The standardized signals will have zero means. The standard deviations of the input

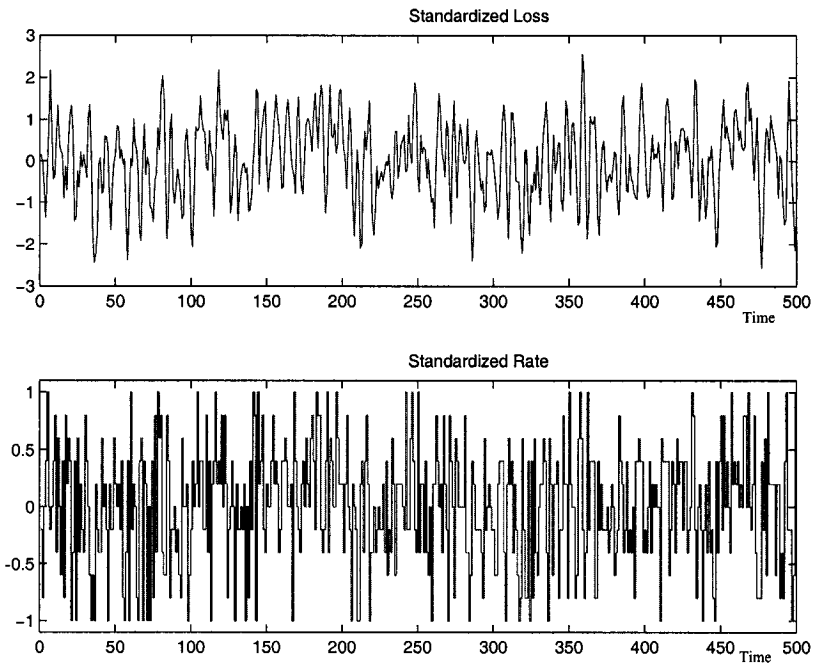


Figure 6.3: Data History for the Packet Rate and Loss

and output signals, denoted by σ (Traffic Rate) and σ (Loss Probability) respectively, are listed in Table 6.1. The first sample is also plotted graphically with Figure 6.3.

Table 6.1: Statistics Synopsis of Traffic Flows

Number	Video Name	σ (Traffic Rate)	σ (Loss Probability)
1	Mutanx	0.5401	0.4437
2	Star War	0.6413	0.4014
3	MTV	0.4413	0.2792
4	Nature	0.3482	0.1495
5	Sports	0.4576	0.3478

6.4 Experimental Result and Model Validation

Software packages are already available for a practical implementation of the identification algorithm. The major burden lies in the calculation of the sequence of prediction errors and the errors' gradients. The matlab [40] routine, *pem*, is used to calculate the estimations in the thesis.

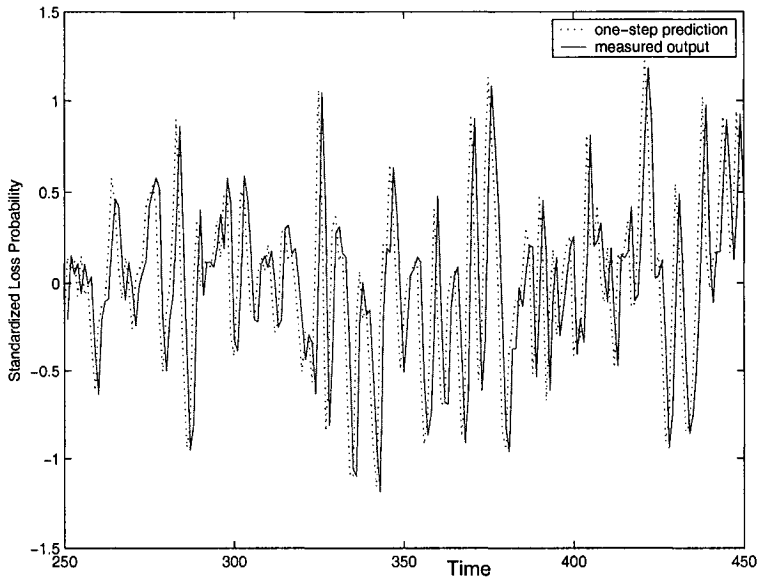


Figure 6.4: Comparison of the Measured and predicted Loss Probability with Order = 2

Based on the observed 200 data from time 51 to 250 in the first sample, the state-space matrixes are estimated in the following Table 6.2:

Table 6.2: Results of the Parameter Identification

$$A = \begin{bmatrix} 0 & 1 \\ -0.323 & 0.618 \end{bmatrix} \quad C = \begin{bmatrix} 0.236 & 0.542 \end{bmatrix}$$

To evaluate the model's quality, the model can be simulated with the actual input and compare its prediction with the actual output. The evaluation is performed on the dataset from time 256 to 450 in the sample one. The result is shown in Figure 6.4, which indicates that the one-step prediction is matched well with the actual output.

6.4.1 Model Order Validation

In the previous experiment, the order of the model is derived from the equations. This experiment challenges whether or not other higher-order models give better prediction results than the selected low-order model.

According to Akaike's theory, in a collection of different models, choose the model

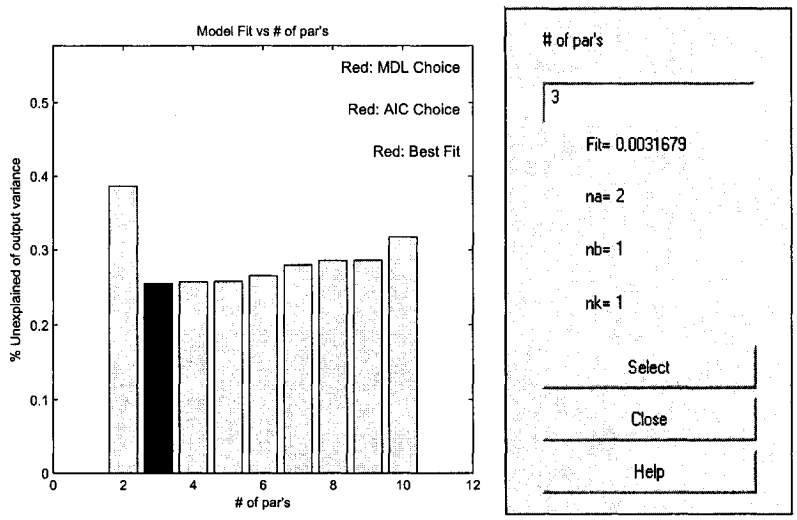


Figure 6.5: Select the best order with Akaike's algorithm

with the smallest FPE or AIC. In this test, first to fifth order models are compared with cross-validation on the second half of the data set. The FPE values are displayed in the following Figure 6.5, which shows the order that gives the best fit to the validation data set. The selected number of model parameters is 3, where $na = 2$ meaning that the dynamics of the model is best described as second order.

It is also natural to compare the results obtained from model structures with different orders. One-step state space predictions are performed with the order from 1 to 6 in the experiments. The model fit results in Table 6.3 show that with the order of 4 it gives the best fit. However, the tabular results also show that the higher-order models do not give a very significant improvement from 86.42% to 86.57%. But in the controlling system, it will be much harder to control a higher order plant. Therefore, in the practical viewpoint, the model with the order of two is the best candidate to identify and control.

Table 6.3: Results of the Prediction with Higher-Order Model

Order	1	2	3	4	5	6
Model Fit (%)	3.82	86.42	86.67	86.71	86.70	86.57

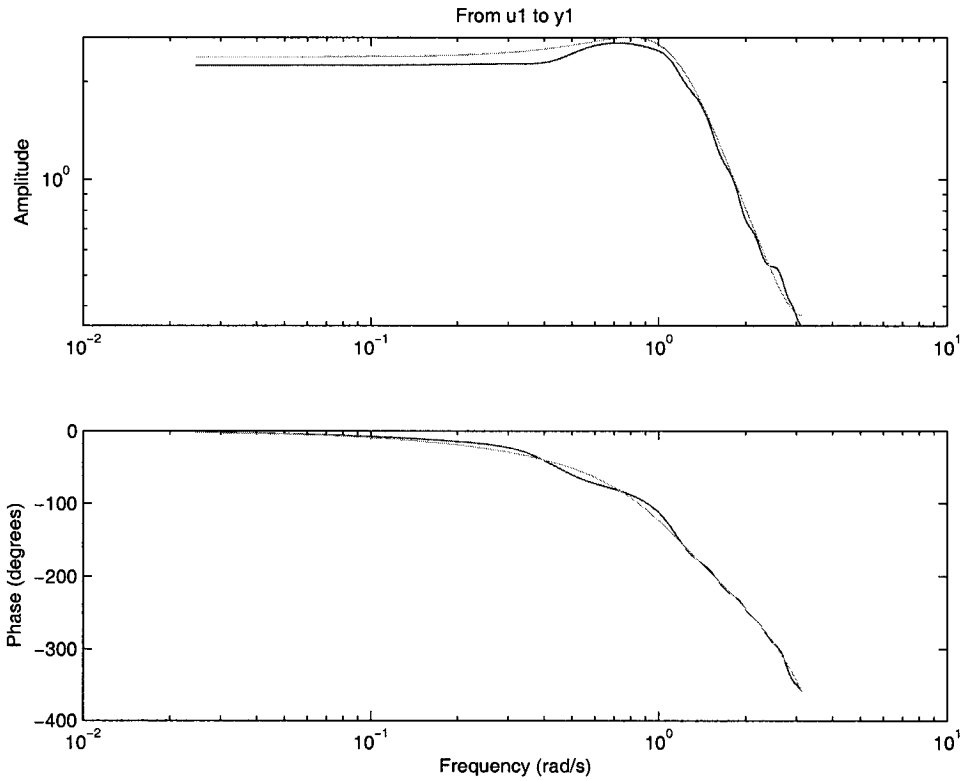


Figure 6.6: Frequency Response of the System

6.4.2 Model Analysis

Once the model has been estimated, its properties should be investigated.

Frequency Response

The frequency response of a linear dynamic model describes how the model reacts to sinusoidal inputs. To know the frequency response, the frequency function of the model can be computed from the identified transfer function. It can also be obtained from a nonparametric, spectral analysis method directly from the measured data. Both of the frequency responses are depicted by two plots in the Bode plot shown in Figure 6.6. The first plot shows the amplitude change as a function of the sinusoid's frequency and the other plot shows the phase shift as a function of frequency.

Comparing the frequency response of the identified transfer function with the response obtained from the nonparametric analysis in Figure 6.6 shows that the agreement is effective.

6.4.3 Parameter Variance

Either a linear approximation of a likely non-linear system or the ignored time-variant properties degrades the identified model. To verify the effectiveness of off-line identification, the model parameters are identified for each of the five data samples listed in Table 6.1. The statistics properties of the identified parameters are evaluated in Table 6.4, which indicates that the model in a different context has a non-ignored time-variance property.

Table 6.4: Parameter Statistics Synopsis

Parameters	Mean	Standard Deviation	Min	Max	Range
α_0	0.590	0.173	0.476	0.721	0.245
α_1	-0.298	0.1681	-0.383	-0.244	0.139
β_0	0.547	0.2391	0.512	0.594	0.082
β_1	0.251	0.184	0.165	0.338	0.173

In Figure 6.7, the analysis of the probability density function also shows that in all cases the parameters have a large variance.

6.5 Recursive Model Identification

The time-variability of the response function and the variable parameters suggest that an adaptive system identification technique might prove useful. Not only would this technique be able to systematically vary model parameters to track non-stationary dynamics in time, but the recursive nature of most adaptive algorithms is ideal for implementation in an on-line operational space model. In the recursive algorithm, the state and parameter estimates are updated without repeated computation of matrix solution that is quite time consuming. Furthermore, the old data is discarded and potential data storage problems are bypassed. In the next experiment, the on-line data is captured and the parameters are identified using the general Recursive Prediction Error Method (RPEM).

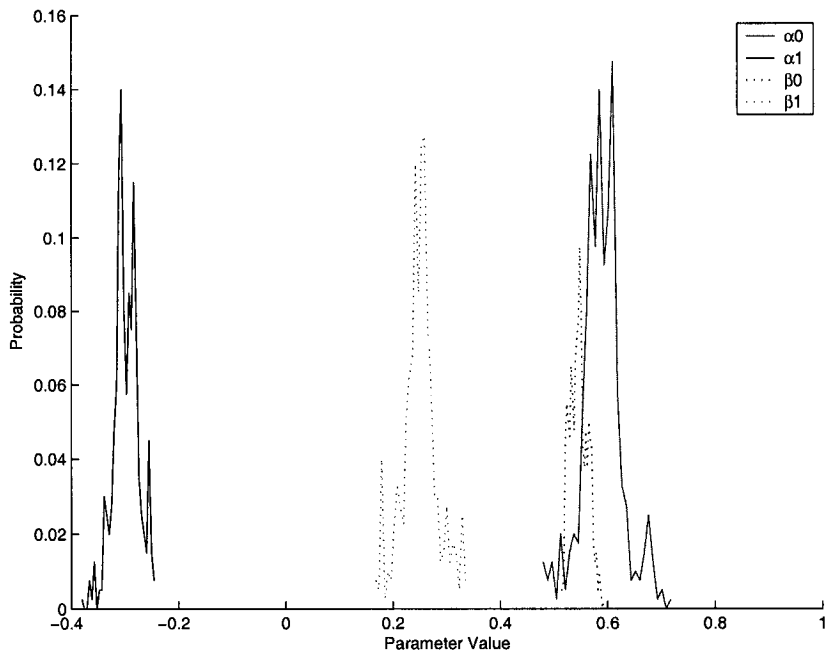


Figure 6.7: Distribution of the parameters.

6.5.1 Recursive Prediction Error Method

The main advantage of the RPEM is its general applicability. The algorithm can be specified in different cases as special names. The RPEM estimates the parameters by minimizing the optimally determined one-step-ahead output prediction error denoted by $\epsilon(k)$. Since information in the latest pair of measurements $(L(k), r(k))$ is normally small compared to the information already accumulated from previous measurements, the algorithm typically utilizes the previous estimated parameters to ensure that the parameters are calculated during a sampling interval. Based on this idea, the algorithm estimates the parameters in a recursive way without repeated computation of matrix solutions that can be quite time consuming.

Analogous to the PEM case, consider a weighted quadratic prediction-error criterion:

$$V(\theta) = \gamma(N)\frac{1}{2}\sum_{k=1}^{N}\beta(k)\epsilon^2(k) \tag{6.39}$$

where $\beta(k)$ is the weight. $\gamma(N)$ is used to normalize the gain and defined as the following:

$$\gamma(N) = [\sum_{k=1}^{N}\beta(k)]^{-1} \tag{6.40}$$

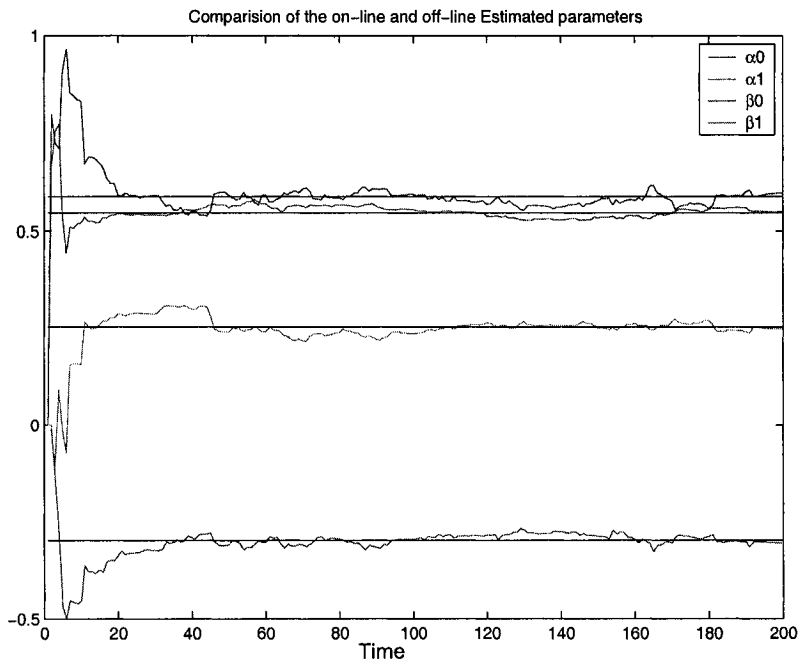


Figure 6.8: Estimated Parameters

Applying the recursive Gauss-Newton prediction-error algorithm [75], the recursive scheme is given by the following group of equations:

$$\epsilon(k) = L(k) - \hat{L}(k) \tag{6.41}$$

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \gamma(k)R^{-1}(k)\psi(k)\epsilon(k) \tag{6.42}$$

$$R(k) = R(k-1) + \gamma(k)[\psi(k)\psi^T(k) - R(k-1)] \tag{6.43}$$

where $R(k)$ is the second derivative of $V(\theta)$ with respect to θ termed the Hessian. $\psi(k)$ is the gradient of the predictions. $\hat{\theta}(k)$ is the estimate of the parameter vector. All of the variables are termed for the k^{th} time interval.

6.5.2 Numerical Results

In the first experiment, the off-line data from the first sample is simulated as on-line data and the parameters are identified using the RPEM algorithm.

The four parameters $(\alpha_0, \alpha_1, \beta_0, \beta_2)$ are plotted as functions of time. To verify the effectiveness of the algorithms, the response time is checked in the experiment. The initial parameter values are set to 0 at the beginning. After a history of about 20, the parameters become stable, which is shown in Figure 6.8.

To calculate the stable error, the four parameter values $(\alpha_0, \alpha_1, \beta_0, \beta_1)$ are compared with the values from the Table I obtained by applying off-line PEM. Both of the parameters match pretty well.

In the next couple of experiments, a long-term on-line data is captured and the parameters are identified using the RPEM algorithm. The captured data in Table 6.1 is also used to calculate the prediction error for the off-line identified model in order to compare the performance. Performances are evaluated by computing the mean of the square of the prediction errors. The result, presented in Table 6.5, suggests that the RPEM approach performs better than the PEM algorithm for long time traffic.

Table 6.5: Comparing the Off-line Approach with the On-line Approach

Experiment	PE^2 (PEM)	PE^2 (RPEM)
1	0.7735	0.0561
2	0.6715	0.0413
3	0.5669	0.0413
4	0.4594	0.0576

The graphic result shown in Figure 6.9 compares the one-step prediction error using the recursively identified parameters and the off-line PEM algorithm. The figure also indicates that the RPEM matches the output better than PEM. That is reasonable becasue the RPEM can adjust itself to the variable context and match the model much better.

6.5.3 Tuning the Forgetting Factor

An important reason for using RPEM is that the properties of the system may be time varying and that the variations should be tracked by the algorithm. This is handled in a natural way in RPEM by assigning less weight to older measurements that are no longer representative for the system. In this, let

$$\beta(k) = \lambda^{(N-k)} \tag{6.44}$$

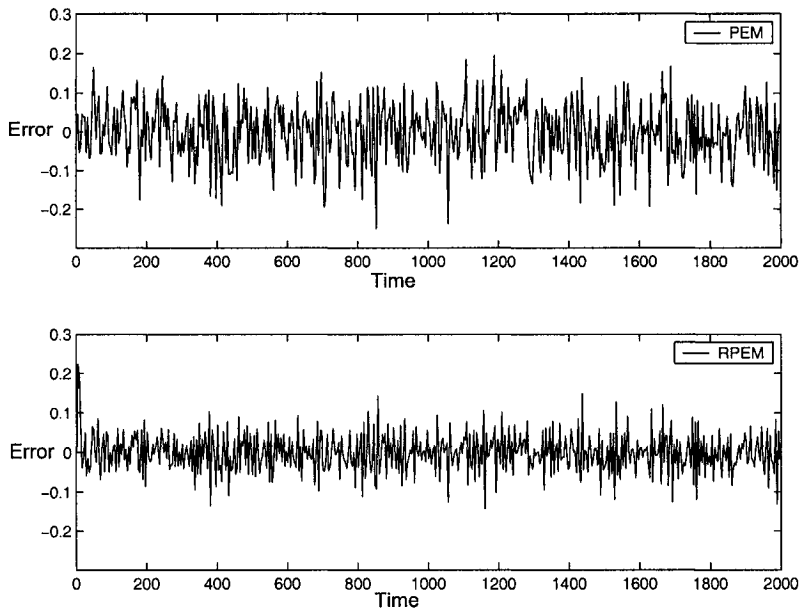


Figure 6.9: Comparison of Prediction Error of PEM with RPEM

where $\lambda < 1$ is often called the forgetting factor. A constant forgetting factor λ gives, after a transient, a constant gain

$$\gamma = 1 - \lambda \tag{6.45}$$

There is always a tradeoff between tracking ability and noise sensitivity. The typical choices of λ are in the range between 0.98 and 0.995. By increasing the value of λ, the algorithm is less sensitive to noise but it can be slower in adaptation. A smaller λ can result in quicker adaptation but the algorithm will be more sensitive to noise as shown in Figure 6.11.

So far in all of the experiments that have been performed, the only value of the RPE method's forgetting factor considered is $\lambda = 0.98$. This is a common value that is used. The forgetting factor can be fine tuned for the process being modeled to obtain improved adaptive parameter estimation.

In the experiment, the values of λ varies between 0.940 and 1.000 with increments of 0.003. The graphic result shown in Figure 6.10 indicates that for most of the processes the optimal value from the set of forgetting factors is 1.0 or very near 1.0.

Different forgetting factors will affect the varying of the parameters. Graphical results are presented for the first sample in Figure 6.11. One may notice that when λ

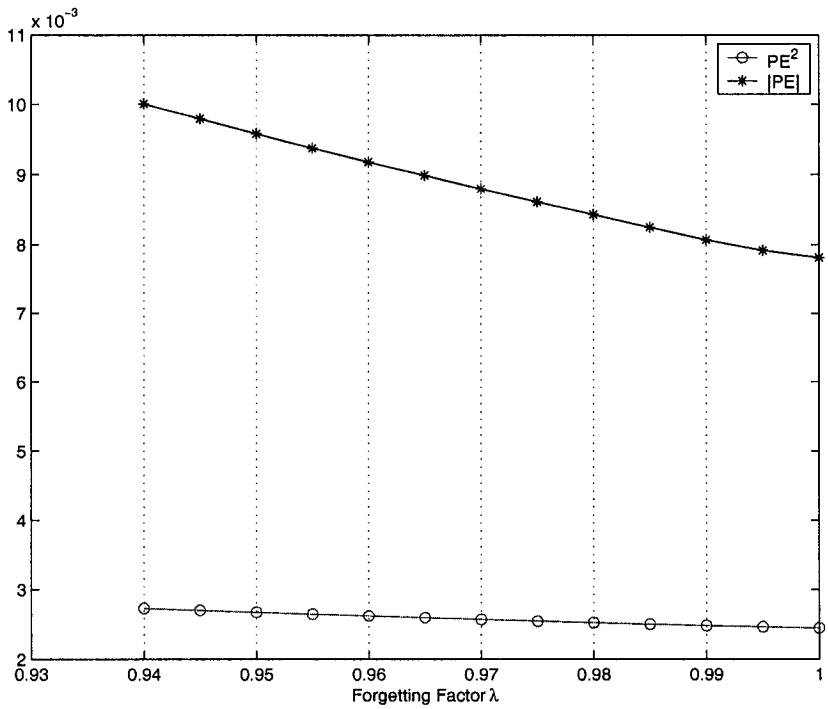


Figure 6.10: Average PE^2 and $|PE|$ with the Varying Forgetting Factor

gets closer to one, the parameter changes very little. On the other hand, when λ equals to 0.940, the parameters are more sensitive to the noise.

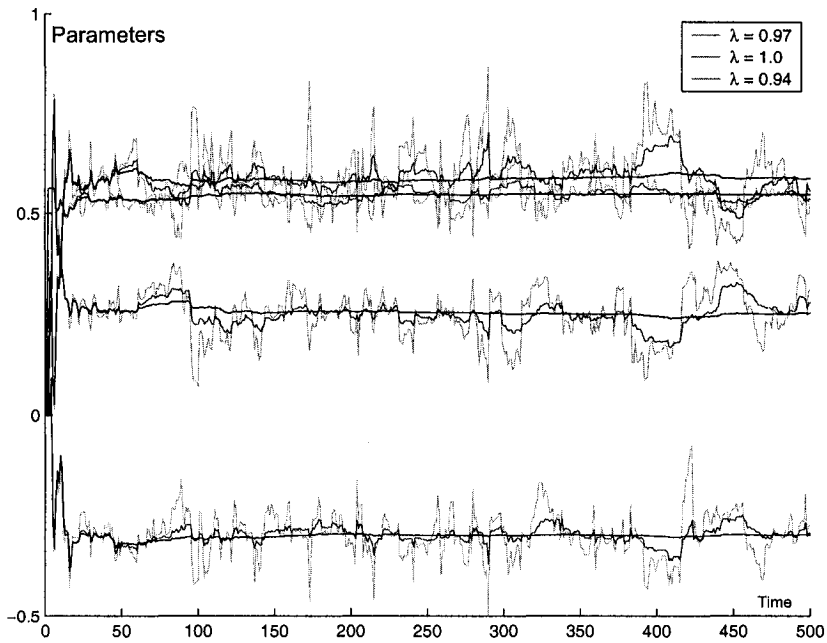


Figure 6.11: Changing of Parameters with Different Forgetting Factors

6.6 Conclusion

The chapter tries to identify the dynamic model for the packet loss system based on the system identification methods, where the ingress packet rates are inputs and packet loss probability is the output. The linear state model is selected and initial parameters are obtained from the offline data. Prediction error method and recursive prediction error method are implemented to estimate the model parameters. The numeric results are evaluated on the live NCIT*net2 network, which verifies the effectiveness of the algorithm. The control of the identified system for guaranteeing QoS is the subject of a next chapter.

Chapter 7

Digital Controller Design

In the preceding chapter, the dynamic packet loss model is identified with the system identification method. This Chapter will try to approach the loss control problem for the identified nominal model. Both the input-output (the classical control theory) approach and the input-state-output (the modern control theory) approach will be investigated and experimented within a real network packet loss control loop.

7.1 Classical Controller Design

7.1.1 Introduction

The design of a control system involves the selection of an appropriate controller or system compensator transfer function and the tuning of the controller parameters so that a quality index of the system is obtained. The desired characteristics or performance specifications, generally includes steady-state error, transient response, disturbance rejection, etc.

Classical design approach usually deals with Single-Input, Single-Output (SISO) system. A simple system of this type is shown in Figure 7.1. For this system, the system transfer function is formulated as:

$$\frac{C(z)}{R(z)} = \frac{D(z)G(z)}{1 + D(z)GH(z)} \tag{7.1}$$

and the characteristic equation is

$$1 + D(z)GH(z) = 0 \tag{7.2}$$

There are various techniques for the controller design in the literature. Some of them are reviewed in the following subsections.

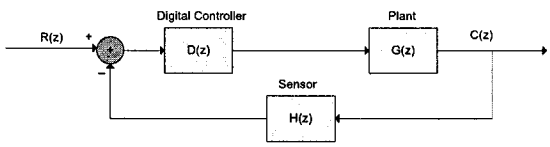


Figure 7.1: Simple Feedback Control System

Frequency Response Analysis and Design

The frequency response method is less intuitive than other methods. However, it has certain advantages, especially in real-life situations such as modeling transfer functions from physical data.

The frequency response is a representation of the system's response to sinusoidal inputs at varying frequencies. The output of a linear system to a sinusoidal input is a sinusoid of the same frequency but with a different magnitude and phase. The frequency response is defined as the magnitude and phase differences between the input and output sinusoids. The frequency response of a system can be viewed in two different ways: via the Bode plot or via the Nyquist diagram. Both methods display the same information; the difference lies in the way the information is presented.

To plot the frequency response, the value of the plant transfer function is computed at a vector of frequencies. If $G(z)$ is the transfer function of a system and ω is the frequency vector, then plot $G(j * \omega)$ vs. ω. Since $G(j * \omega)$ is a complex number, the Bode diagram plots both its magnitude and phase while the Nyquist diagram plots the position of the complete function $G(j * \omega)$ vector in the complex plane.

In the frequency-response design procedures, phase-lead or phase-lag compensators attempt to reshape the system open-loop frequency response to achieve certain stability margins, transient-response characteristics, steady-state response characteristics and so on. Even though the design equations can be developed, the techniques are still largely of the type trial and error. Moreover, the method of frequency response is less intuitive.

Root Locus

Root locus for a system is a plot of the roots of the system's characteristic equation as gain is varied. A generic example is shown in Figure 7.2, where two roots are selected

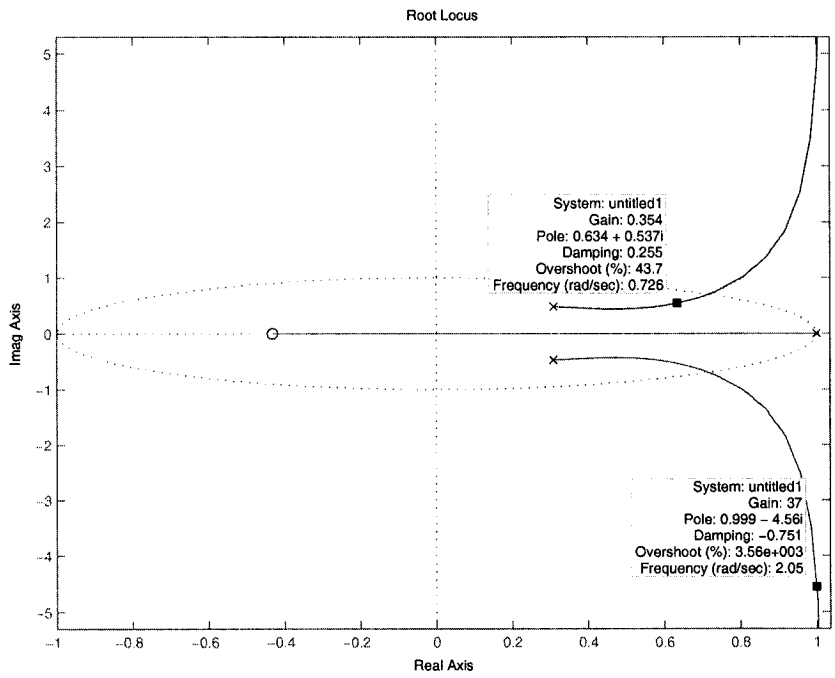


Figure 7.2: A Generic Example of Root Locus for A Feedback System

and the correspond system properties are also shown. In this way, the character of the transient response of a system is judged based on values obtained from the root locus. The design procedure is to add poles and zeros via a digital controller so as to shift the roots of the characteristic equation to more appropriate locations in the z-plane. In the following section, root locus technique is employed to design the controller for the packet loss system.

7.1.2 Packet Loss Controller Design via Root Locus

The approach via root locus involves iteration on a design by manipulating the compensator gain in the root locus diagram.

Design Procedure

The packet loss control system is shown in Figure 7.3, where $L_0(z)$ is the reference packet loss input, $L(z)$ is the actual packet loss probability. The controller $D(z)$ is designed with root locus method. When the packet loss probability is higher than the preset value, the ingress traffic will be throttled with no more than 1% per customer, which

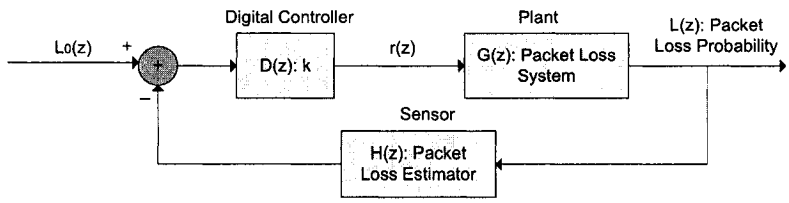


Figure 7.3: Diagram of the Control System Design with Root Locus

will bring the core traffic within the existing core bandwidth. On the other hand, if the packet loss probability is lower than the preset value, this means that the customer VPN flows do not use the full bandwidth assigned to them. As the action of any control system take place around the functional point in a narrow band, the bandwidth for the ingress traffic will be loosed with a very small percentage. On one hand, this will bring more benefits to customers while on the other hand, the customers do not need extra bandwidth as they do not have traffic for using it. Practically the controller action will not have any effect on the existing traffic flows and therefore on the SLA. A final remark is related to the fact that the action of the controller in this case is limited on the upper side by the customer contract, and thus a negative variation of the PLP might not lead to any action of the controller. This latter fact can be implemented in a very simple logic in the controller software.

The packet loss model, $G(z)$, is identified with the system identification method from previous chapter, where the state-space model of the packet loss system is described as:

$$\begin{bmatrix} q_1(k+1) \\ q_2(k+1) \end{bmatrix} = A \begin{bmatrix} q_1(k) \\ q_2(k) \end{bmatrix} + Br(k) \tag{7.3}$$

$$l(k) = C \begin{bmatrix} q_1(k) \\ q_2(k) \end{bmatrix} + Dr(k) + e(k) \tag{7.4}$$

where $\begin{bmatrix} q_1(k) \\ q_2(k) \end{bmatrix}$ is the vector of states, $l(k)$ is the output, $r(k)$ is the input and $e(k)$ is the measurement noise, in the k^{th} time interval.

The model structure is identified by the following four matrixes in the canonical form:

$$A = \begin{bmatrix} 0 & 1 \\ \alpha_0 & \alpha_1 \end{bmatrix} B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} \beta_0 & \beta_1 \end{bmatrix} D = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

There are four parameters, α_0, α_1, β_0 and β_1, which characterize the proposed model, are identified in the experiments following the procedure described in Chapter 6.

In the buffer control system proposed by Meerkov [10], the dynamic buffer system is assumed as a first-order equation:

$$x_i(n+1) = Sat_x\{x_i(n) + f_i(n) - c\}, i \in \mathcal{N} \tag{7.5}$$

where $x_i(n)$ is the average buffer size of buffer i, $f_i(n)$ is the conditional expectation of the input process to link i (number of packets that arrive during time slot $[n, n+1)$ with distribution:

$$P_{\varsigma_i}(n,k) = Prob\{\varsigma_i(n) = k\}$$

and

$$Sat_a(z) = min[(0,z)^+, a]$$

where $[x]^+ = max(0,x)$.

In the classical design viewpoint, the plant transfer function, $G(z)$, is seen to be:

$$GH(z) = \frac{L(z)}{r(z)} \tag{7.6}$$

$$= C[zI - A]^{-1}B + D \tag{7.7}$$

$$= [\beta_0 \ \ \beta_1] \begin{bmatrix} z & -1 \\ -\alpha_0 & z - \alpha_1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{7.8}$$

$$= \frac{\beta_0 + \beta_1 z}{z^2 - \alpha_1 z - \alpha_0} \tag{7.9}$$

where α_0, α_1, β_0 and β_1 are identified parameters.

In the packet loss control system shown in Figure 7.3, the sensor, $H(z)$, is the packet loss estimator, which estimates the packet loss probability based on the measurement of the traffic and is constructed in Chapter 5. If the input $L_0(z)$ is the reference packet

loss probability from the SLA and consider the system shown in Figure 7.3, $D(z)$ is changed to a scalar gain k to be adjusted when the one-parameter root locus design is employed. The system is formulated as:

$$\frac{L(z)}{L_0(z)} = \frac{kG(z)}{1 + kGH(z)} \tag{7.10}$$

In such a scheme, the closed-loop poles are the roots of the characteristic equation (7.11):

$$1 + kGH(z) = 0 \tag{7.11}$$

where k is the added gain that is to be varied to generate the root locus.

The root locus for a system is a plot of the roots of the system's characteristic equation as gain is varied. The design procedure is to tune the loop gain in order to shift the roots of the characteristic equation to more appropriate locations in the z-plane.

Combined with equation (7.9), the characteristic equation (7.11) is changed to (7.14)

$$1 + k\frac{\beta_0 + \beta_1 z}{z^2 - \alpha_1 z - \alpha_0} \tag{7.12}$$

$$= \frac{z^2 - (\alpha_1 - k\beta_0)z - \alpha_0 + \beta_1 k}{z^2 - \alpha_1 z - \alpha_0} \tag{7.13}$$

$$= 0 \tag{7.14}$$

The root locus technique consists of plotting the closed-loop pole trajectories in the z plane as k varies. This plot is used to identify the gain value associated with a desired set of closed-loop poles.

7.1.3 Design Criteria

In the controller design process, different design criteria results in different controller parameters. In this section, the controller is designed for the packet loss system, so that the damping ratio of the closed loop system is set to be an appropriate value, for example 0.7.

The following Figure 7.4 shows the root locus of the system in the z-plane, where the grid is formulated by the two arguments: natural frequency (ω_n) and damping ratio (ζ). The poles in z-plane is located at

$$z = r\angle \pm \theta \tag{7.15}$$

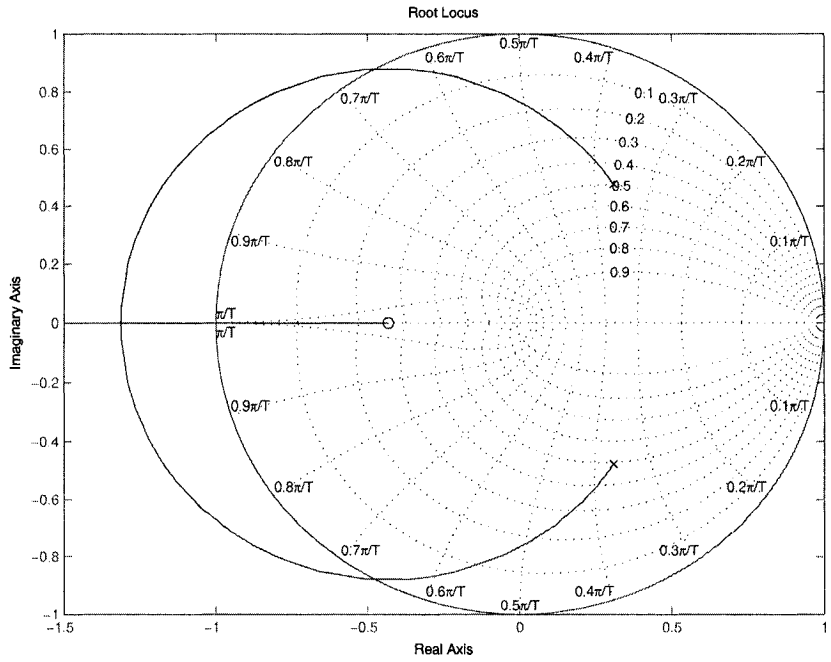


Figure 7.4: Diagram of the Root Locus for the Packet Loss System

The z-plane poles occur at

$$z = e^{-\zeta\omega_n T}\angle \pm (\omega_n T\sqrt{1-\zeta^2}) \tag{7.16}$$

where T is the time interval in discrete time.

Therefore, the damping ratio and natural frequency are expressed in poles in the z-plane as:

$$\zeta = \frac{-lnr}{\sqrt{ln^2 r + \theta^2}} \tag{7.17}$$

$$\omega_n = \frac{1}{T}\sqrt{ln^2 r + \theta^2} \tag{7.18}$$

Clicking on the locus, some design parameter is selected and the corresponding system characteristic is calculated by the Matlab. By selecting appropriate point on the root locus, for example $k = 2.3$, the following design specification is achieved in Table 7.1 according to the equations (7.17) and (7.18).

The following Figure 7.5 plots the closed-loop response with the designed controller and compares it to the closed-loop impulse response without the controller. The closed-loop response with the controller settles down quickly and does not oscillate too much.

Table 7.1: Design Specification

Gain:	2.304
Pole:	0.578+0.232i
Damping Ratio:	0.789
Maximum Overshoot (%):	2.01
Natural Frequency (rad/sec):	122

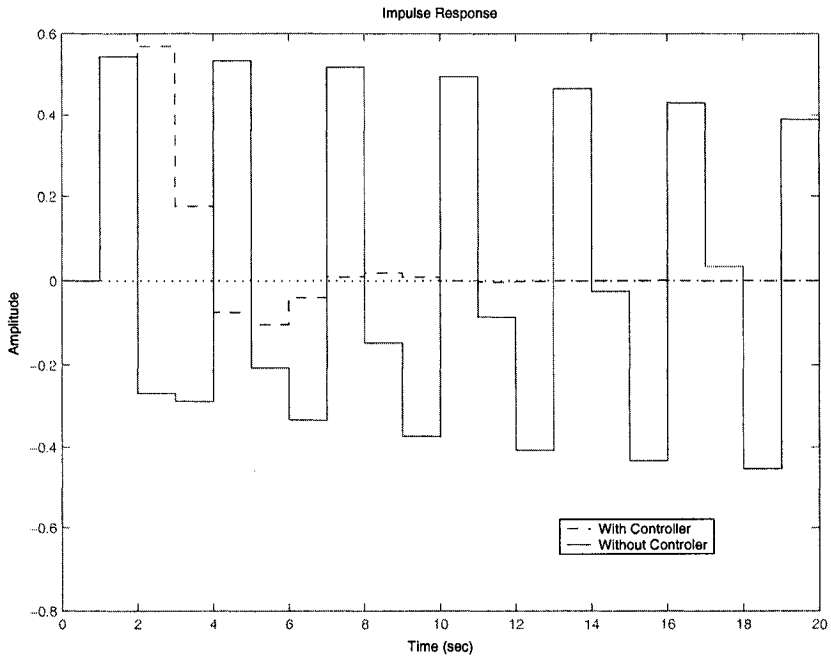


Figure 7.5: Diagram of Loss Response for the Packet Loss System

7.1.4 Experimental Result

To verify the effectiveness of maintaining the loss target in real traffic, a number of experiments are designed on the test bed shown in Figure 4.3. The following traffic shown in Figure 7.6 is generated and measured on the Eth1/2 of Cisco1 through the management workstation. The traffic is then used for simulation on the matlab to verify the effectiveness of the control algorithm.

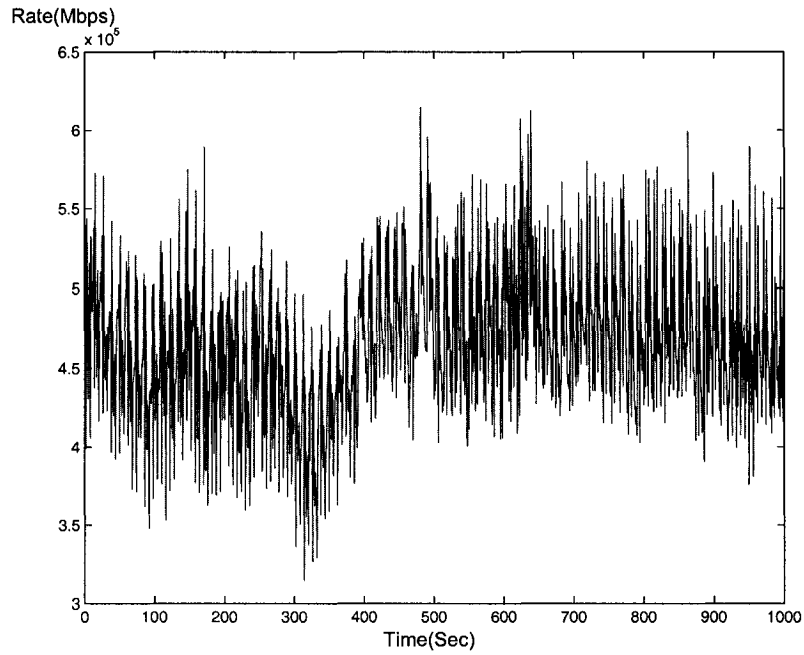


Figure 7.6: Diagram of Real Traffic Measured on Eth1/2 on Cisco1 in NCCT Network

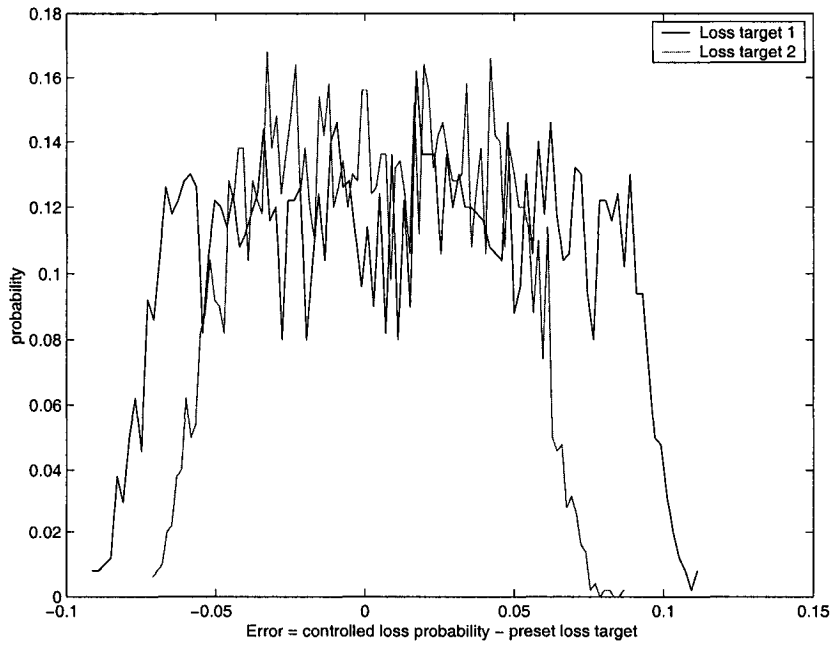


Figure 7.7: PDF of the Error between the Measured Loss Ratio and Preset Loss Target with Classical Controller on Eth1/2 on Cisco1 in NCCT Network

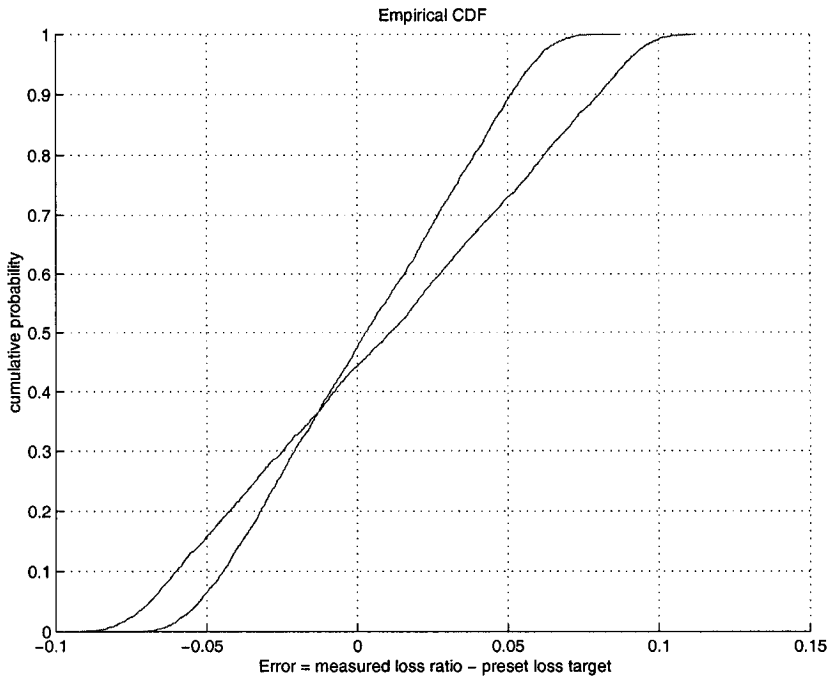


Figure 7.8: CDF of the Error between the Measured Loss Ratio and Preset Loss Target with Classical Controller on Eth1/2 on Cisco1 in NCCT Network

The errors between the measured loss ratio and the preset loss target are recorded for all of the experiments. In Figure 7.7, the analysis of the probability density functions shows that in all cases the control system keeps the actual loss close to the target loss. Figure 7.8 shows the cumulative probability density function, from which one can see that there is a steady error but that the expectation of the error is very close to zero. About 90% of the errors are in the region [-0.125, 0.125]. Since the target of the packet loss control system is to maintain the packet loss under the preset level, the small steady errors show the effectiveness of the control system.

7.2 State Variable Approach to the Controller Design Problem

In the last section, the classical transfer function approach is employed to represent the transfer function of the controller approached by root locus method. However, the systems with classical control approach must be linear and time invariant, and have a single input and a single output. That is, only one signal has to be fed back. It seems reasonable that if more information about the states of a system can be used to generate the control input, the control of the system will be of a better quality [92].

The basic characteristic of the state-variable formulation is that linear and nonlinear systems, time-invariant and time-variant systems and single-variable and multivariable systems can all be modeled in a unified manner. The classical transfer function representation is shown in Figure 7.9 (a), where only one input and one output can be represented by the transfer function. The system is modeled in state-variable representations and is shown in Figure 7.9 (b), which allows for the possibility of more than one input and output.

In Figure 7.9 (b), the variables $u_i(k)$ are the external inputs which drive the system, while the variables $y_i(k)$ are the system output or the system response. The variables $x_i(k)$ are the internal or state variables of the system, which represent the minimum amount of information to determine both the future states and the system outputs for given input functions. That is, given the system states, the system dynamics and the input functions, all subsequent states and outputs are determined.

The state-space model of the packet loss system is first derived as an IIR system in

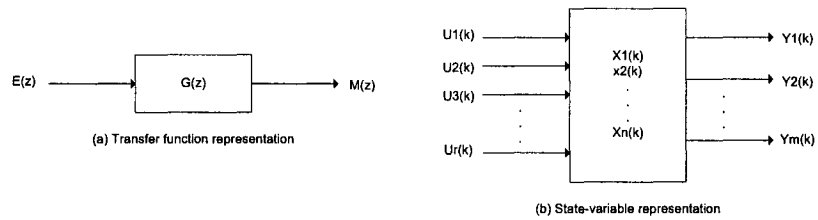


Figure 7.9: Representations of System Dynamics

Chapter 6.

$$\begin{bmatrix} q_1(k+1) \\ q_2(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \alpha_0 & \alpha_1 \end{bmatrix} \begin{bmatrix} q_1(k) \\ q_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(k) \tag{7.19}$$

$$l(k) = \begin{bmatrix} \beta_0 & \beta_1 \end{bmatrix} \begin{bmatrix} q_1(k) \\ q_2(k) \end{bmatrix} + e(k) \tag{7.20}$$

where $\begin{bmatrix} q_1(k) \\ q_2(k) \end{bmatrix}$) is the vector of states, $l(k)$ is the output, $r(k)$ is the input and $e(k)$ is the measurement noise in the k^{th} time interval.

¿From these equations, two state variables can be written as:

$$q_1(k+1) = q_2(k) \tag{7.21}$$

$$q_2(k+1) = \alpha_0 q_1(k) + \alpha_1 q_2(k) + r(k) \tag{7.22}$$

The two state variables, $q_1(k)$ and $q_2(k)$ can be assumed as the buffer size in the last two time intervals, respectively. The buffer sizes are also assumed to be measurable. Compared with the buffer system proposed by Meerkov [10], where the $\alpha_0 = 0$, the dynamic buffer system in this thesis describes a more general case.

7.2.1 Controller Design via Pole Assignment

A design procedure generally known as pole assignment [92] is developed to control the state-variable model in this section.

The admission rates are calculated in the control algorithm to be a linear combination of the states. That is,

$$r(k) = -\sum_{j=1}^{J} K_j q_j(k) \tag{7.23}$$

To study the asymptotic stability properties, the dynamic equation is simplified by removing the saturation nonlinearities. If let $J = 2$, (7.23) will be simplified as:

$$r(k) = -K_1 q_1(k) - K_2 q_2(k) \tag{7.24}$$

Then system can be written as:

$$q(k+1) = \begin{bmatrix} 0 & 1 \\ \alpha_0 & \alpha_1 \end{bmatrix} q(k) - \begin{bmatrix} 0 \\ 1 \end{bmatrix} [k_1 q_1(k) + k_2 q_2(k)] \tag{7.25}$$

that can be rewritten as:

$$q(k+1) = \begin{bmatrix} 0 & 1 \\ \alpha_0 - k_1 & \alpha_1 - k_2 \end{bmatrix} q(k) \tag{7.26}$$

where the closed-loop system matrix is named A_f.

$$A_f = \begin{bmatrix} 0 & 1 \\ \alpha_0 - k_1 & \alpha_1 - k_2 \end{bmatrix} \tag{7.27}$$

and then the characteristic equation will be:

$$|zI - A_f| = 0 \tag{7.28}$$

After some calculations, the characteristic equation can be simplified as:

$$z^2 + (k_2 - \alpha_1)z + k_1 - \alpha_0 = 0 \tag{7.29}$$

Suppose that two roots, λ_1 and λ_2 are chosen according to some design specifications. Then the desired characteristic polynomial is given by

$$(z - \lambda_1)(z - \lambda_2) = z^2 - (\lambda_1 + \lambda_2) + \lambda_1 \lambda_2$$

Equating the coefficients, yields the equations

$$k_2 = -(\lambda_1 + \lambda_2) + \alpha_1 \tag{7.30}$$

$$k_1 = \lambda_1 \lambda_2 + \alpha_0 \tag{7.31}$$

Thus, the gain matrix $K = [k_1 \;\; k_2]$ can be found that will realize any desired characteristic equation. In the next experiment, the pole place is chosen to satisfy design criteria such as speed of response, overshoot in the transient response, and the like.

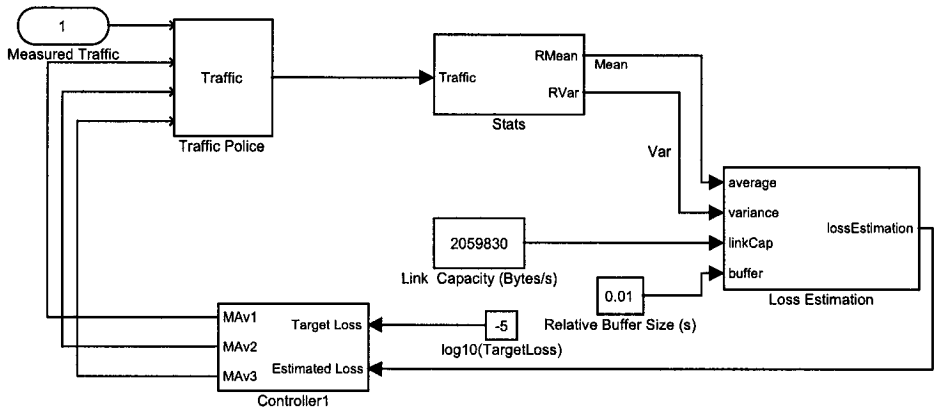


Figure 7.10: Implementation Diagram of the Control System in Matlab

7.2.2 Controller Implementation

A number of experiments were designed and implemented on the live test bed shown in Figure 4.3. The traffic is measured and packet loss probability is estimated by applying the algorithm introduced in Chapter 4 and 5. The measured live traffic is then applied as the input to the control system simulated within Matlab system, where the controller is implemented to verify the effectiveness of the control system.

The simulation diagram of pole-assignment based control system is shown in Figure 7.10. Three PE nodes are simulated in the test, where the nodes send traffic out. The measured traffic mean and variance are sent to loss estimation process, where the estimated loss is calculated and sent to the controller. The controller generates the admissible rates for each PE node and those rates are sent back to the three PE nodes.

In order to demonstrate the system dynamics, the live traffic is measured on the Eth1/2 of Cisco1 through the management workstation and one sample of the traffic is shown in Figure 7.11. The traffic is then used for the input to the Matlab program to verify the effectiveness of the control algorithm.

7.2.3 Performance of the Controller

The loss target is preset in the SLA. As an example, it is set to be 10% in this experiment. At time 300 and 600, the packet loss is sharply increased with the increasing of the traffic shown in Figure 7.11.

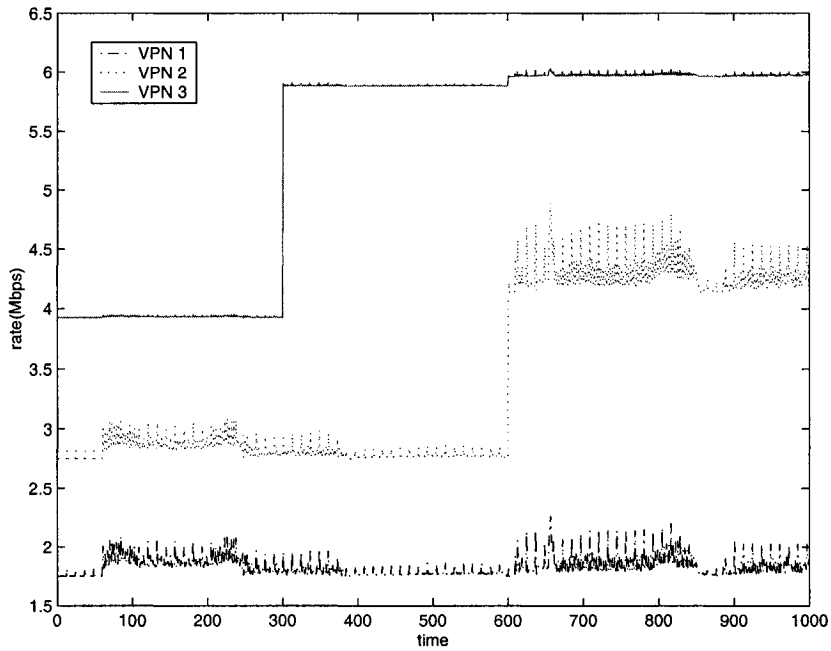


Figure 7.11: Input Packets of the Three VPN Traffic Measured on Eth1/2 on Cisco1 in NCCT Network

Since the loss target is preset to 10% and before time 300 the packet loss probability is below the target, it is not necessary to activate the control algorithm. That is, all of the user traffic can enter the core network. After time 300, the packet loss is increased to 20%, so the loss control system should take actions to control the traffic rates in order to decrease the packet loss probability to 10% again. The same policy should be taken after time 600.

Parameter Selection

In the following experiments, different root locations (λ_1, λ_2) are chosen so as to satisfy design criteria such as speed of response, maximum overshoot in the transient response, damping ratio and the like. In the following experiments, two sets of root locations (λ_1, λ_2) are detailed in Table 7.2 to check how the system dynamic behavior varies.

When the first set of control parameters in Table 7.2 is selected, the performance is shown in Figure 7.12. It is seen that the packet loss probability is controlled to be down back to the preset value (10%) after the congestion happened. The response time is about 50 time periods. With such a set of gains, the system is stable. There is a

Table 7.2: Two Sets of k_1 and k_2

	k_1	k_2
1	0.8	0.01
2	1.0	0.005

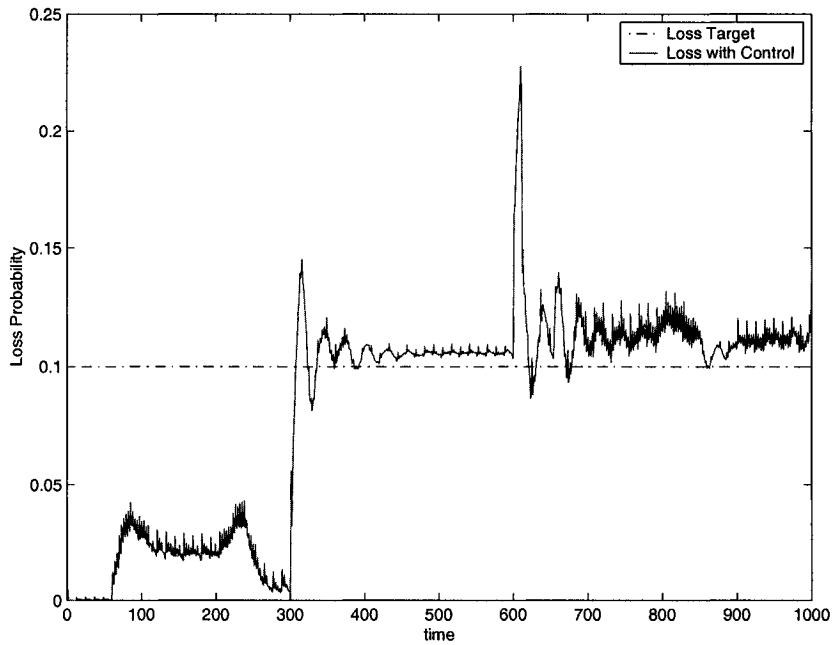


Figure 7.12: Controlled Packet Loss Probability with the First Set of Parameters

steady error, but it is very small.

Transient Performance Analysis

When the gain k_1 is continually increased to the second parameter set in Table 7.2, the steady error decreases to zero. But the system becomes more unstable shown in Figure 7.13. After the first congestion, the system cannot converge. For the second congestion, the system can be converged, but the response time is very long. As it is shown, after the time 1000, the system still does not reach the stable state.

Steady State Error

To verify the effectiveness of maintaining the loss target in a steady state, a number of experiments are designed, and the errors between the controlled loss probability and the preset loss target are recorded and evaluated. Two loss targets are set in the following

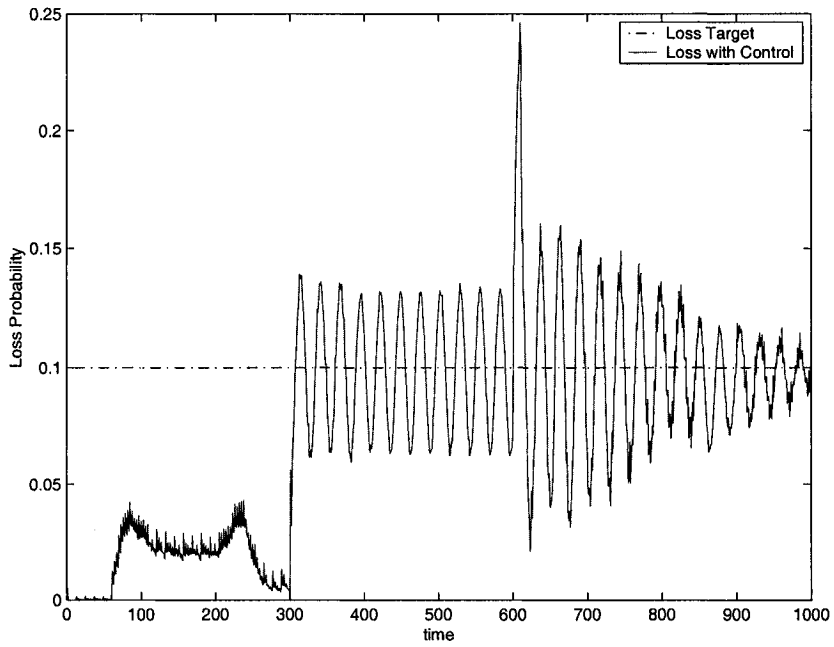


Figure 7.13: Controlled Packet Loss Probability with the Second Set of Parameters

experiments: one is 10% and the other one is 15%. With these two loss targets, a large number of experiments are designed by varying the input traffic rates.

In Figure 7.14, the analysis of the probability density function shows that the control system can keep the actual loss close to the target loss in all cases. Figure 7.15 shows the cumulative probability density function, from which one can see that there is a control error but that the expectation of the error is very close to zero. About 90% of the errors are in the region [-0.05, 0.05], which shows the effectiveness of the control system. The small error region also proves that state variable based controller has a better performance than the previous classical controller.

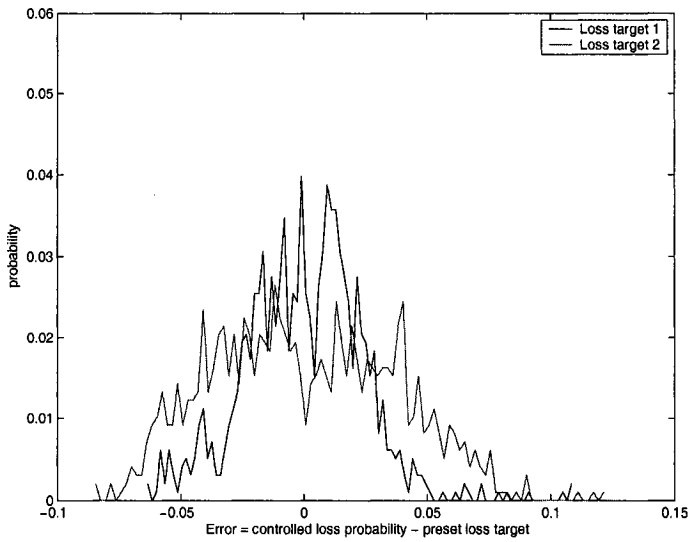


Figure 7.14: PDF of the Error between the Measured Loss Ratio and Preset Loss Target with Modern Controller Designed in Section 7.2

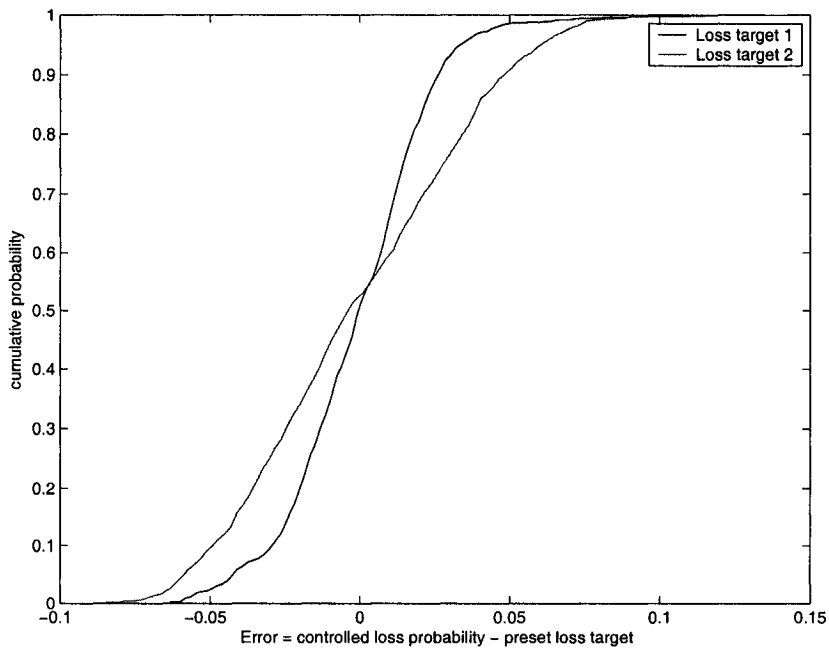


Figure 7.15: CDF of the Error between the Measured Loss Ratio and Preset Loss Target with Modern Controller Designed in Section 7.2

7.3 Conclusion

The chapter presents a control theory approach for the controller design for the control loop of the packet loss management for admitted VPN services in the provider's backbone network. The system tries to maintain the preset packet loss probability in the core network towards avoiding the congestion and providing long-term statistical QoS for the customers. The experimental results show that the proposed controller can maintain the loss value within a range of preset target, if appropriate parameters are selected. However, the identified model has variable parameters. A robust and optimal controller design technique is required to deal with the uncertain model. That will be investigated in next chapter.

Chapter 8

Robust and Optimal Control

In the previous chapter, the controller is designed for the identified nominal model to throttle the customers' ingress traffic. The classical input-output and more modern input-state-output control technologies have been studied. The classical input-output methods are effective, but are largely based on trial and error, while experience is very important. Even when an acceptable design is completed, the question remains as whether or not a better design could be found. The pole assignment design technique is one modern technique, where the exact pole location can be specified for the closed loop equation. However, the precise pole location is another unsolved issue in the design of the packet loss control loop. Of course, there are many others. However, this pole-assignment technique assumes that the exact pole locations that yield the *best* control system are known.

In this chapter, a different technique, known as optimal design technique that yields the *best* control system, is investigated. An optimal design technique minimizes the predefined cost function and satisfies the requirements in a best way. Thus there is no need to know a-priori the position of the poles.

Another issue is that the identified linear model for the packet loss system contains some uncertain parameters when the parameters are identified by applying the method proposed in Chapter 6. The uncertainty is caused by the linearization process (i.e. the probability of packet loss process is not linear and is affected by a random variation in the process parameters). Therefore, the digital controller should be designed to perform satisfactorily over the anticipated range of plant parameter variations. The design technique is know as *Robust control*, which makes the closed-loop system continue

performing satisfactorily despite large variations in the plant dynamics.

However, robustness trades with performance. In this chapter, a controller is designed with best performance for specified plant-parameter robustness, where parameters are identified by the system identification method and varied in a certain interval. A Linear Matrix Inequality (LMI) approach for designing the Proportional-Integral (PI) controller for the identified dynamic system is studied [99, 48]. Firstly, the robust PI control design problem is formulated as LMIs. By adopting the LMI expression, a symmetric positive definite matrix P with guaranteed overall system's quadratic stability can then be easily determined. The matrix P can finally be used to infer the robust PI controller parameter. The transient and steady state performance of the controller is evaluated on the live NCIT*net2 network with a number of experiments. The numeric results show that, under different traffic type, it is possible to maintain the network operation robustly within a prescribed performance.

8.1 Uncertainty and Robustness

8.1.1 Uncertainty of the Identified Model

In Chapter 6, it has been done to identify the dynamic packet loss model with a RPE method. To be practical, the packet loss system is simplified as a linear time-invariant model in order that it is convenient for the beginning of the model analysis. The state-space uncertain model of the packet loss system is derived as:

$$q(k+1) = (A + \delta A)q(k) + Br(k) \tag{8.1}$$

$$l(k) = (C + \delta C)q(k) + Dr(k) + e(k) \tag{8.2}$$

where $q(k)$ is the vector of states, $l(k)$ is the output, $r(k)$ is the input and $e(k)$ is the measurement noise in the k^{th} time interval; A, B, C,D are nominal constant matrices of appropriate dimensions and δA, δC are matrices of uncertainties of appropriate dimensions.

To simplify the analysis further, the system is assumed as a second order model, where the initial structure can be described with the following four matrixes:

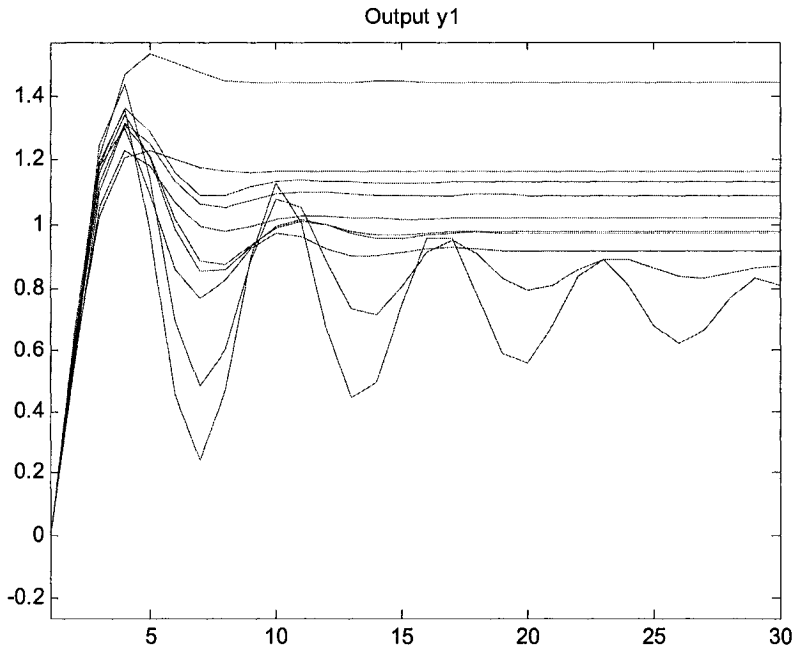


Figure 8.1: Step Response of Identified Uncertain Model

$$A = \begin{bmatrix} 0 & 1 \\ \alpha_0 & \alpha_1 \end{bmatrix} B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} \beta_0 & \beta_1 \end{bmatrix} D = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

There are four parameters, α_0, α_1, β_0 and β_1, which are identified in the experiments. As the result of the RPE method, the identified linear model contains norm-bounded uncertain parameters. Therefore, δA and δC are random matrixes. The step response of the identified model is shown in the following Figure 8.1, which visually shows the uncertainty of the system.

The term *uncertainty* refers to the differences or errors between models and reality [134]. To be practical, the identified system model cannot exactly describe the physical system. The inevitable inadequacy exists in all of the models. In order to be applicable in practice, the model is highly simplified; otherwise it would be too complicated to handle for a controller design procedure. Therefore, the controller design is based on a crude mathematical model of the physical reality.

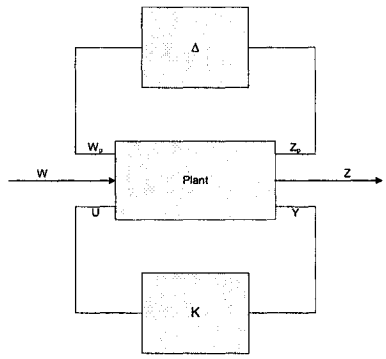


Figure 8.2: General Framework for an Uncertain Feedback System

There are two basic types of uncertainty: parameter uncertainty and unstructured uncertainty. Representing uncertainty is usually to separate the uncertainty from the plant by creating an additional block which is connected to the plant by feedback. The general uncertain feedback system is depicted in the following Figure 8.2

In this chapter, only the parameter uncertainty is dealt with, since the identified model has uncertain parameters. δA and δC are matrices of uncertainties and considered to be of the affine type:

$$\delta A = \sum_{j=1}^{p} \epsilon_j A_j$$

$$\delta C = \sum_{j=1}^{p} \epsilon_j C_j$$

where $|\epsilon_j| < 1$ are unknown parameters; A_j, C_j, $j = 1, 2, ..., P$, are constant matrices of the corresponding dimensions. The affine parameter can be readily converted into a polytopic model and described by a list of its vertices:

$$(A_1, C_1), (A_2, C_2), ..., (A_N, C_N), N = 2^P \tag{8.3}$$

8.1.2 Robust Stability Analysis

Testing stability of a system is very important in linear system analysis and synthesis. The Lyapunov theory is useful for testing the system stability. In control theory, the discrete Lyapunov equation is a system of the form:

$$A^T X A - X + Q = 0 \tag{8.4}$$

where A and Q are given real matrices. The discrete Lyapunov equation can also be written as:

$$\begin{bmatrix} X & XA \\ A^T X & X - Q \end{bmatrix} = 0 \tag{8.5}$$

The continuous Lyapunov equation is expressed as:

$$AX + XA^T + Q = 0 \tag{8.6}$$

For a linear system described by $x(k+1) = Ax(k)$, the relationships between the stability of A and the solution of the Lyapunov equation are provided by the following lemma [134].

Lemma 8.1 Assume that A is stable, then the following statements hold:

$$(i)\ Q = \int_0^\infty e^{A^T t} H e^{At} dt$$

$$(ii)\ Q > 0\ if\ H > 0\ and\ Q \geq 0\ if\ H \geq 0$$

$$(iii)\ If\ H \geq 0,\ then\ (H, A)\ is\ observable\ \ iff\ Q > 0$$

The consequence of this lemma is that, given a stable matrix A, a pair (C, A) is observable if and only if the solution to the following Lyapunov equation (8.7) is positive definite.

$$A^T X + XA + C^T C = 0 \tag{8.7}$$

In applications, the solution of the Lyapunov equation is calculated and the stability of matrix A should be concluded according to the following Lemma 9.2.

Lemma 8.2 Suppose X is the solution of Lyapunov equation (8.7), then the following statements hold:

$$(i)\ Re\lambda_i(A) \leq 0\ if\ X > 0\ and\ C^T C \geq 0$$

$$(ii)\ A\ is\ stable\ if\ X > 0\ and\ C^T C > 0$$

$$(iii)\ A\ is\ stable\ if\ X \geq 0,\ C^T C \geq 0,\ and\ (C^T C, A)\ is\ detecatable$$

where $Re(\alpha)$ means the real part of the complex number α, $\lambda_i(A)$ is the i^{th} eigenvalue of matrix A.

Therefore, the identified packet loss system, shown by (8.1) and (8.2), is asymptotically stable iff there exists X, so that $X > 0$ and $A^T X + XA + C^T C < 0$. Since the δA

and δC are considered to be of the affine type, the stability is whether all trajectories of the system converge to 0 time $k \rightarrow \infty$. A necessary and sufficient condition [18] for this is that there exists a variable matrix P to satisfy all of the following inequalities:

$$P > 0,\ A_i^T P + P A_i + C_i^T C_i < 0, i = 1, ..., m \tag{8.8}$$

To solve the Linear Matrix Inequalities (LMI) efficiently, some methods are introduced in the next section.

8.2 Linear Matrix Inequality

A Linear Matrix Inequality (LMI) [18] is any constraint of the form

$$A(x) := A_0 + \sum_{i=1}^{n} x_i A_i < 0 \tag{8.9}$$

where $x \in R^n$ is the variable vector to be solved and the symmetric matrices $A_i \in R^{n \times n}$, i = 0,...,n are given constant matrices. The inequality (< 0) stands for negative definite, i.e., the largest eigenvalue of $A(x)$ is negative. Eigenvalues are a special set of scalars associated with a linear system of equations that are sometimes also known as characteristic roots. On the other hand, a positive definite matrix is a Hermitian matrix where all eigenvalues are positive. For example, the three positive definite 2x2 matrices are:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

all of which have eigenvalue 1.

The importance of the LMI arises in system and control theory because the controller design can be formulated as convex optimization problems with LMIs that are amenable to computer solution. The LMI is proved to be a convex constraint on x from the following equation:

$$A(y) < 0,\ A(z) < 0 \Longrightarrow A(\frac{y+z}{2}) < 0 \tag{8.10}$$

As a result, finding a solution x^* to (8.9), is a convex optimization problem. Convexity has an important consequence: even though (8.9) has no analytical solution in general, it can be solved numerically with guarantees of finding a solution when one exists, which has great practical importance.

There are three generic LMI problems:

- Feasibility problem: find a solution x^* to the LMI system $A(x) < 0$, or to determine that no such x^* exists.
- Linear objective minimization problem: Minimizing a convex objective under LMI constrains, such as Minimize $C^T x$ subject to $A(x) < 0$.
- Generalized eigenvalue minimization problem: minimize λ, subject to $A(x) < \lambda B(x), B(x) > 0$ *and* $C(x) < D(x)$.

All of the three problems can be solved in polynomial-time [17]. There are lots of commercial software package that solve the LMI problems. Among them, the LMI LAB within the Control Toolbox [40] implements state-of-the-art interior-point LMI solvers. The interior-point methods by Nesterov [82] are extremely efficient and are significantly faster than classical convex optimization algorithms. Their high performance is achieved through C-MEX implementation and by taking advantage of the particular structure of each LMI. C-MEX files are functions written in C, which can run on any system that is already running MATLAB.

Even though the canonical expression in (8.9) is generic, LMIs rarely arise in this form in control application, since this expression is less intuitive and transparent. Moreover, the number of matrices grows roughly as $n^2/2$ if n is the size of the A matrix, so that the canonical form is very inefficient from a storage viewpoint. According to the reasons above, the LMI Lab uses a structured representation of LMIs.

In general, LMIs assume a block matrix form where each block is an affine combination of the matrix variables.

$$L(X,\gamma) = \begin{bmatrix} A^T X + XA & XC^T & B \\ CX & -\gamma I & D \\ B^T & D^T & -\gamma I \end{bmatrix} < 0 \tag{8.11}$$

where $X \in R^{n\times n}$ and $\gamma \in R$ are the matrix variables of the problem. A, B, C, D are given matrices. $L(X,\gamma)$ is a symmetric block matrix. Each block of $L(X,\gamma)$ is an affine expression in the matrix variables X and γ.

8.3 Controller Design

8.3.1 Linear Quadratic Optimal Control

The optimal design procedure minimizes the cost function and therefore yields a best control system. Consider the linear dynamic system (8.1)and (8.2), the cost function is generally the form:

$$J_N = \sum_{k=0}^{\infty} L[l(k),\ r(k),\ u(k)] \tag{8.12}$$

Usually a quadratic cost function of the states and the control signals is studied in the optimal control literature, since the development is simple and the cost function is logical in practice. This technique is named Linear Quadratic Regulator (LQR).

The standard LQR problem is to determine the control law u of the form:

$$u(k) = -Kx(k) \tag{8.13}$$

which minimizes the quadratic cost:

$$J(u) = \sum_{0}^{\infty}(x^T Qx + u^T Ru) \tag{8.14}$$

for any initial state x(0), where Q and R are symmetric positive semi definite matrix and positive definite matrix, respectively. It turns out that the solution u^* to this optimal control problem can be expressed in the state feedback form:

$$u^* = -Kx = -R^{-1}B^T Px \tag{8.15}$$

where P is the symmetric positive definite solution of the Algebraic Riccati Equation (ARE):

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{8.16}$$

and the minimum quadratic cost is given by

$$J_{min} = x^T(0)Px(0) \tag{8.17}$$

The solution to the LQR problem relies on solving the ARE equation (8.16). An efficient alternative for this problem is the LMI technique due to this convexity. By the LMI technique, the LQR problem can be rephrased as an optimization problem over P and Y:

$$Minimize\ x^T(0)Px(0) \tag{8.18}$$

subject to

$$\begin{bmatrix} AP^{-1} + P^{-1}A^T + BY + Y^TB^T & P^{-1} & Y^T \\ P^{-1} & -Q^{-1} & 0 \\ Y & 0 & -R^{-1} \end{bmatrix} \leq 0 \tag{8.19}$$

where $P^{-1} > 0$ and $Y = R^{-1}B^T$.

In the practice situation, the objective (8.18) is modified as

$$x^T(0)Px(0) \leq \gamma \tag{8.20}$$

where γ is the specified upper bound. This inequality can also be expressed as an LMI:

$$\begin{bmatrix} \gamma & x^T(0) \\ x(0) & P^{-1} \end{bmatrix} \geq 0 \tag{8.21}$$

Consequently, the optimization problem is converted to seek a solution (P^*, Y^*) that satisfies both of LMIs in (8.19) and (8.21). Then, the state feedback gain is given by:

$$K = -Y^*P^* \tag{8.22}$$

8.3.2 Robust Controller Design

The advantage of LMI technology is its convenience to include other specifications for the controller design, since the controller is usually required to meet various specifications simultaneously in practical situations. Various design specifications may be recast into the LMIs and the resulting LMI constraints can be solved in polynomial time by recently developed convex optimization algorithms.

In this thesis, a PI controller is designed in the state-space setting with LQR-LMI approach. A PI controller has the following transfer function:

$$D(z) = K_p + K_i\frac{z+1}{z-1} \tag{8.23}$$

In the state-space model, the PI controller design becomes a static state feedback controller, and the static feedback gain K simply contains all the PI controller parameters.

For the design specification, it is well known that the transient response of the dynamic system is closely related to the locations of poles. The desired damping can be achieved by constraining the closed loop pole to lie in a prescribed region. The conditions for the pole placement constrain are also recast in the LMIs and LMI region.

Let $L = L^T = [\lambda_{i,j}]_{1\leq i,j\leq m}$ and $M = M^T = [\mu_{i,j}]_{1\leq i,j\leq m}$, which means that L and M are mxm matrix with the elements λ_{ij} and $\mu_{i,j}$, respectively. The closed-loop poles lies in the LMI regions:

$$Z \in \mathcal{C} : L + Mz + m^T \bar{Z} < 0 \tag{8.24}$$

if and only if there exists a symmetric positive definite matrix $\hat{P}$ satisfying the following LMI[25]:

$$[\lambda_{jk} + \mu_{jk}(A\hat{P} + BY) + \mu_{kj}(A\hat{P} + BY)^T]_{1\leq j,k\leq m} < 0 \tag{8.25}$$

The robust PI controller design procedure is shown as the following steps:

- First, a set of multiple models is derived from the identified uncertain model.
- Set the LQR weighting matrix Q and R
- Find the solution (P^*, Y^*) to LMIs (8.19), (8.21) and (8.25)
- Calculate the PI controller parameters by (8.22)

8.4 Experimental Result

In this section, the same traffic flow in previous chapter is used to illustrate the proposed robust and optimal PI controller design method; the performance is compared with the conventional controller in last section. Based on the measured traffic data, the identified system model parameters are as follows:

The nominal matrix:

$$A = \begin{bmatrix} 0 & 1 \\ -0.323 & 0.618 \end{bmatrix} \quad C = \begin{bmatrix} 0.236 & 0.542 \end{bmatrix}$$

The interval values for the four uncertain parameters are:

$$\alpha_0 \in [0.476\ 0.721], \quad \alpha_1 \in [-0.383\ -0.244]$$
$$\beta_0 \in [0.512\ 0.594], \quad \beta_1 \in [0.251\ 0.338]$$

The vertices of the corresponding polytopic model are:

$$A_1 = \begin{bmatrix} 0 & 1 \\ -0.383 & 0.476 \end{bmatrix} \quad C_1 = \begin{bmatrix} 0.251 & 0.512 \end{bmatrix}$$

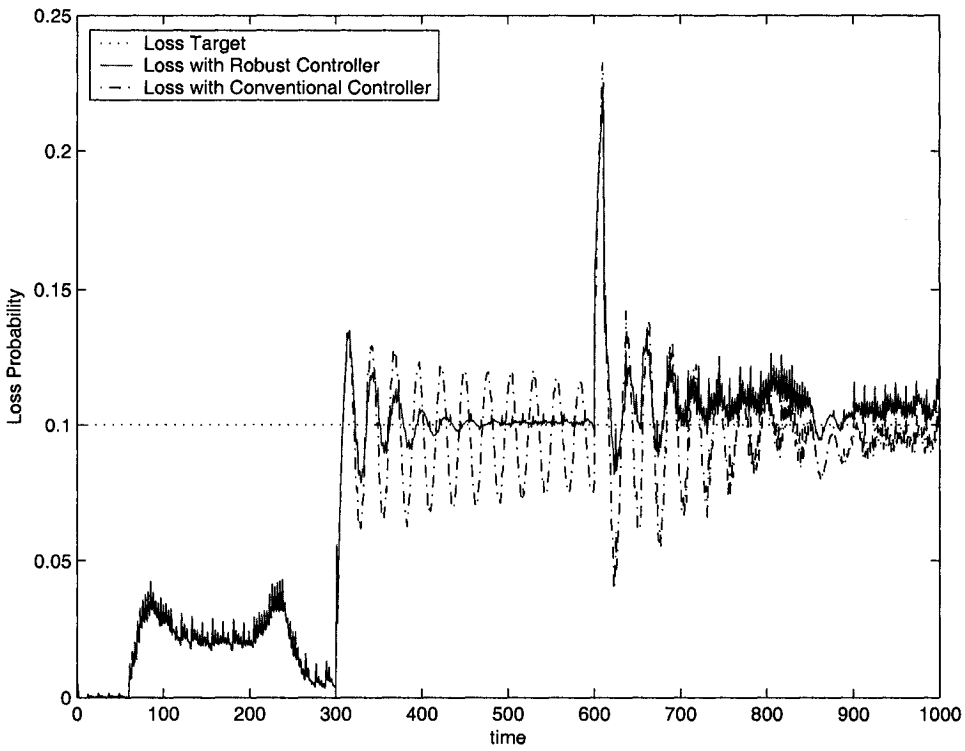


Figure 8.3: Closed Loop Responses to Step Packet Loss Change on the Eth1/2 of Cisco1 in NCCT Network

$$A_2 = \begin{bmatrix} 0 & 1 \\ -0.383 & 0.721 \end{bmatrix} \quad C_2 = \begin{bmatrix} 0.251 & 0.594 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0 & 1 \\ -0.244 & 0.476 \end{bmatrix} \quad C_3 = \begin{bmatrix} 0.338 & 0.512 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} 0 & 1 \\ -0.244 & 0.721 \end{bmatrix} \quad C_4 = \begin{bmatrix} 0.338 & 0.594 \end{bmatrix}$$

Specify the LQR cost weight as $Q = 1$, $R = 1$, $\gamma = 200$ as one of the examples. By the proposed method, the PI controller parameters are obtained as:

$$K = [K_p,\ K_i] = [0.301,\ 0.505] \tag{8.26}$$

For comparison, simulation results are also presented for the conventional controller designed in the previous section, where the control gain is the following:

$$K = [K_p,\ K_i] = [0.252,\ 0.668] \tag{8.27}$$

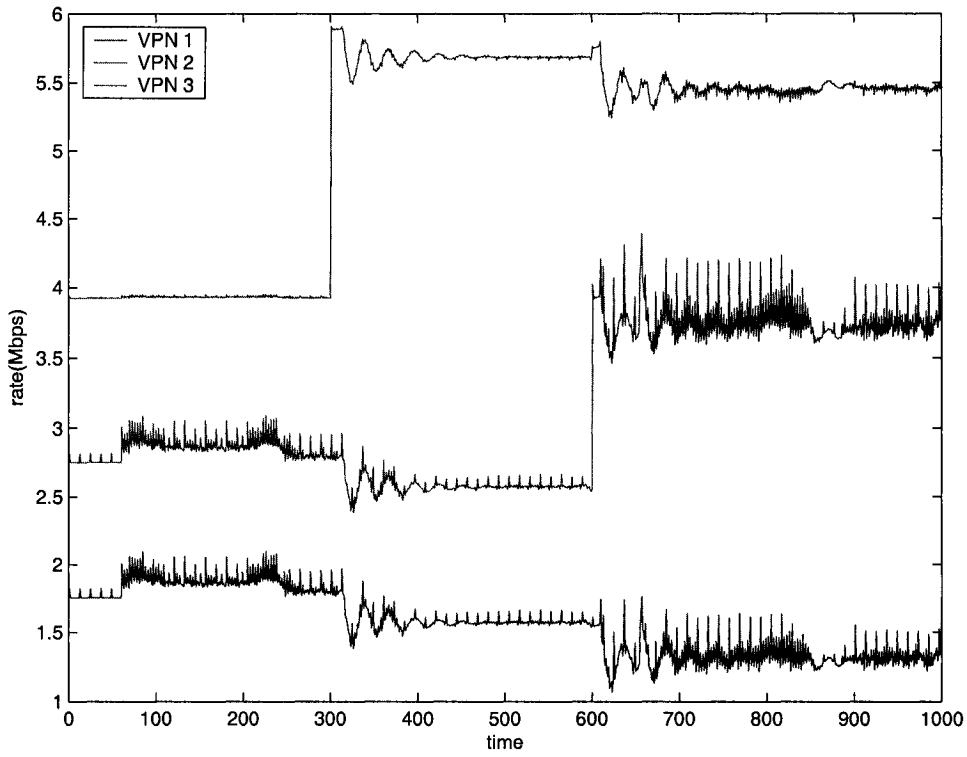


Figure 8.4: Controlled Rates for Each Flow on the Eth1/2 of Cisco1 in NCCT Network

The step-response performance is shown in the Figure 8.3 and the controlled rates for each VPN flow are shown in the Figure 8.4. From the figure, it is observed that the robust controller performs better than the conventional controller globally when there are successive step changes. This can be justified since the robust PI controller has considered the uncertainties of the system in the design procedure so that the controller shows strong robustness in the whole operating region.

To verify the effectiveness of maintaining the loss target in variable conditions, different traffic is injected and long history data are measured. The errors between the controlled loss probability and the preset loss target are recorded and evaluated. Two loss targets are set in the following experiments: one is 10% and the other one is 15%. With these two loss targets, a large number of experiments are designed by varying the input traffic rates.

In Figure 8.5 (I), the analysis of the probability density function shows that the control system can keep the actual loss close to the target loss in all cases. Figure 8.5

(II) shows the cumulative probability density function, from which one can see that there is a control error but that the expectation of the error is very close to zero. About 90% of the errors are in the region [-0.02, 0.04], which shows the effectiveness of the control system.

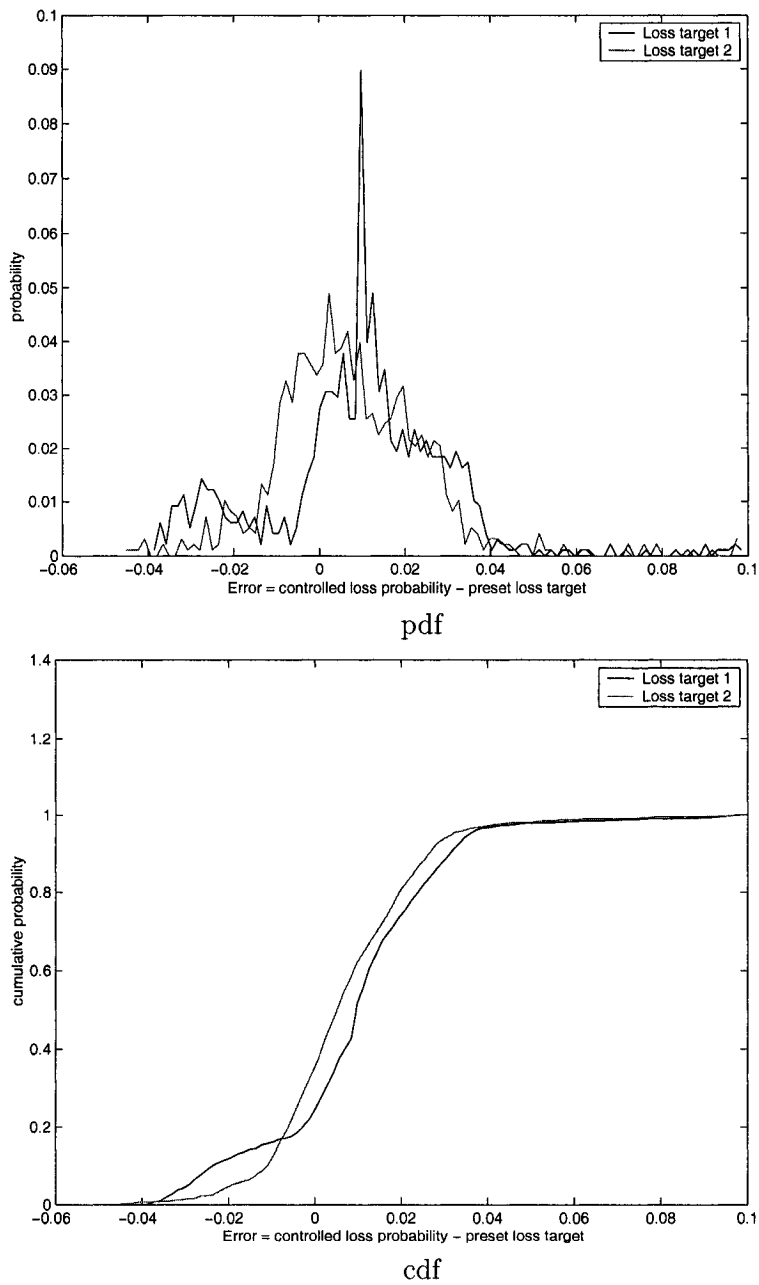


Figure 8.5: Distribution of the Error between the Measured Loss Ratio and Preset Loss Target on the Eth1/2 of Cisco1 in NCCT Network

Chapter 9

Complete Dynamic QoS Control System

In this chapter, the main results of the research developed by the author of this thesis are assembled in a complete closed loop control system. The whole feedback control system is constructed based on previously introduced components. Specifically, a Kalman filter based reactive estimator of the packet loss probability is employed as the transducer for the system, a RPE method is used to identify the parameters of the dynamic packet loss system, and a robust controller is designed for conducting the controller poles in an adaptive manner to keep the packet loss probability in desired limits. The traffic is measured from the live NCIT*net2 network, and is then transferred into the Matlab program for processing.

9.1 Architecture of the Dynamic Control System

The dynamic QoS control system is proposed for MPLS VPN services, especially for managing the core networks operated by network service providers. The control system is used to manage the MPLS VPN services in a way to enhance their competence with the additional QoS properties.

A typical MPLS VPN network is shown in Figure 9.1, where different Virtual Routing and Forwarding (VRF) is configured for each single VPN service on the PE node. VRF is a technology included in IP network routers that allows multiple instances of a routing table to exist in a router and work simultaneously. NSPs take advantage of VRF to create separate VPNs for customers.

The control system will see the core network as a dynamic packet loss plant and apply

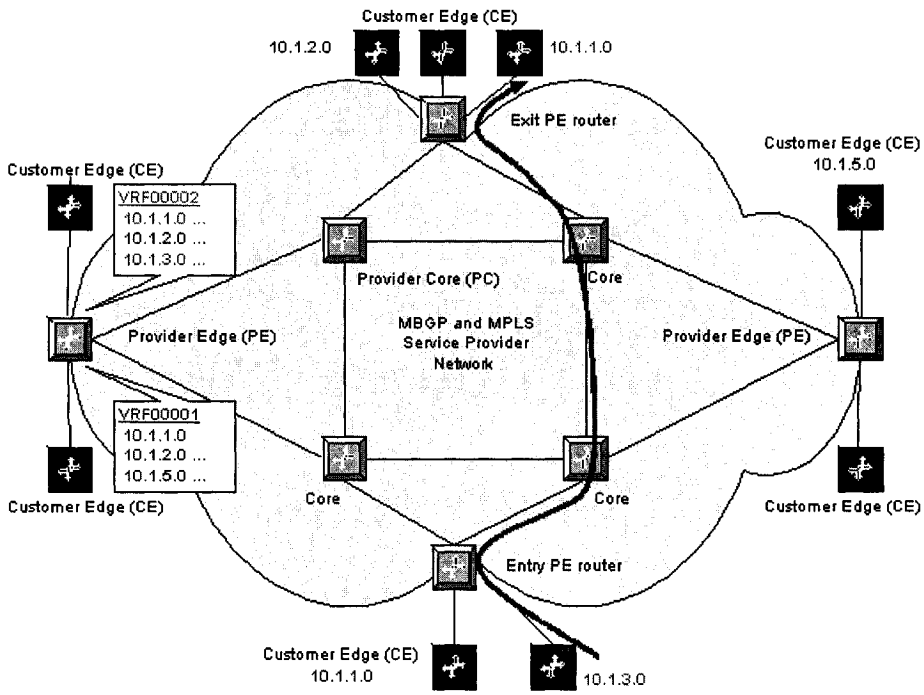


Figure 9.1: One Example of MPLS VPN Network

the control theory to adjust the explicit rate on the PE node in order to meet the QoS requirements for the provisioned VPN services. Unlike the ATM cells, IP packets do not contain control information. Therefore, the system is run on a separated workstation, which builds some connections to the P and PE nodes in the core network. The system measures the traffic from those nodes and sends back the explicit rate signals.

Networks that attempt to deliver more data than their capacity will experience congestion, leading to undesirable data loss, excessive delays, or both. The closed-loop nature of congestion control implies the mechanisms draw heavily on the feedback control theory. The block diagram of the dynamic packet loss control system from a control perspective is depicted in Figure 9.2. The proposed Quality of Service (QoS) control is a process by which networks use feedback to adjust the influx of data so that the customer's QoS requirements are met in the core network while simultaneously attempting to maximize the utilization of the network's resources.

The system consists of three basic components: Packet Loss Measurement and Estimation, Model Identification and Controller Design. Since in most statistical QoS

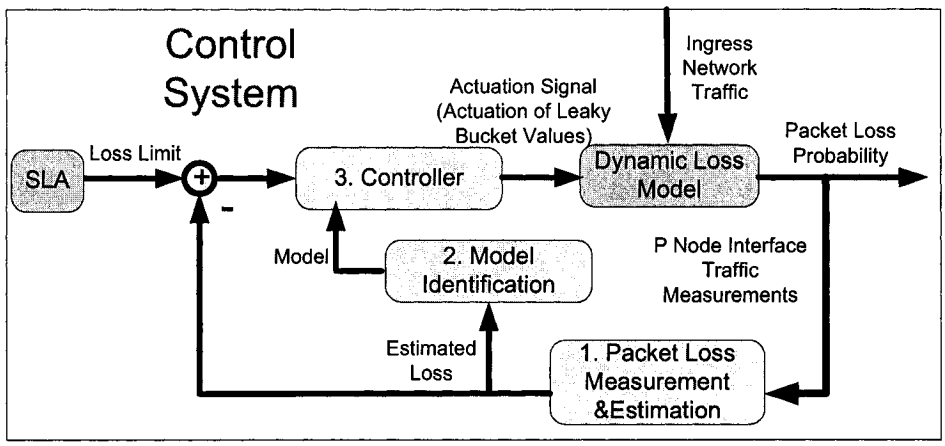


Figure 9.2: Block Diagram of the Packet Loss Control System

environments the packet loss probability is the premier QoS parameter, the packet loss probability is directly used as the feedback signal in this thesis. By throttling the ingress traffic rate in the PE nodes, the packet loss probability in the core network is maintained below the preset value according to the SLA. The procedure is as follows:

1. Packet Loss Measurement and Estimation. To dynamically control congestion and maintain the packet loss in the provider's network, the traffic in the P node is firstly monitored. Then, the measured traffic statistics (average and variance) are sent to the packet loss estimation process, which estimates the packet loss probability according to the equations in Chapter 4.

2. Model Identification. Based on the measured traffic and the estimated packet loss probability, the dynamic packet loss model is identified online through RPE method introduced in Chapter 6.

3. Robust Controller. The digital robust controller is designed based on the identified model. The estimated loss probability and preset loss target are sent to the controller, where the maximum rate for each VPN service is calculated. These explicit rate signals are adjusted by the rate limit actuator process and then sent to the PE node through CLI commands. These commands are executed and take effect on the PE node to throttle the VPN input rates. It is assumed that there is the traffic policer or shaper on the PE node.

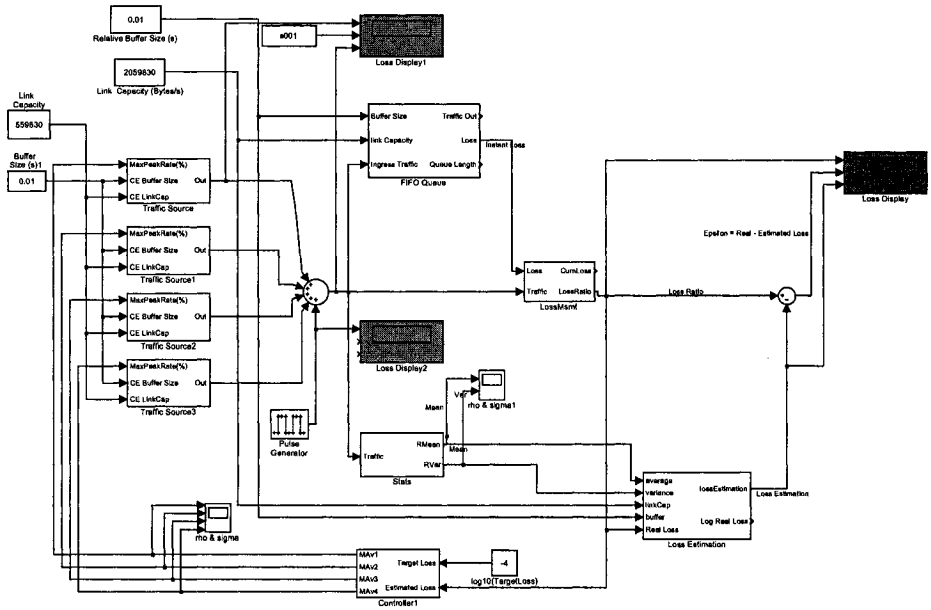


Figure 9.3: Simulation Diagram of the Control System in Simulink

As a proof of concept, the whole control system is implemented in Matlab Simulink, while the input traffic is measured from the live NCIT*net2 network shown in Figure 3.7 in Chapter 3.

9.2 System Design

The control system is implemented in Matlab Simulink to prove the concept. The implemented overall system is shown in Figure 9.3.

Four PE nodes are considered in the demo system, which are connected to one P node. The traffic is measured from the P nodes and the calculated signals are then send back to the four PE nodes. The components in the system are implemented in the following sections, respectively.

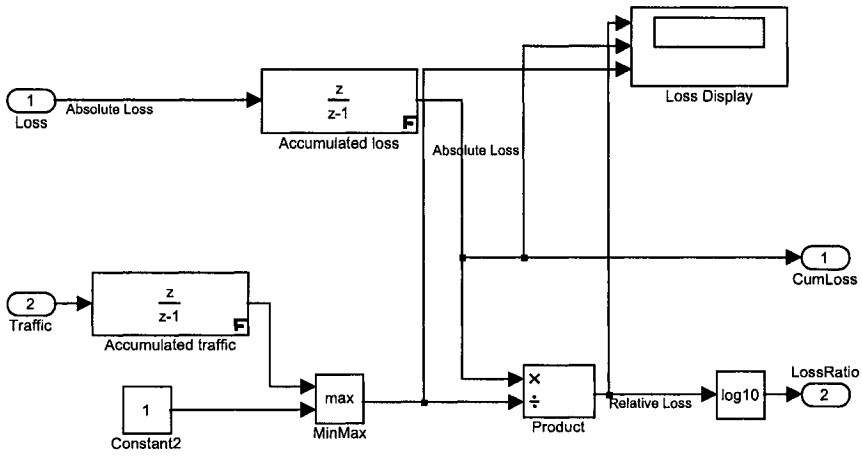


Figure 9.4: Simulation Diagram of the Calculation of the Packet Loss Ratio in Simulink

9.2.1 Transducer

In order to improve the accuracy and maintain applicability in a live network, a *Reactive Estimator* (RE), which can be adaptive to the dynamic traffic model, is constructed as the transducer of the control system.

Assume the k^{th} real-time measured pack loss ratio ($plr(k)$)is available according to the following measurement and calculation:

$$plr(k) = log(\frac{\sum_{i=0}^{R-1} \hat{l}(k-i)}{\sum_{i=0}^{R-1} \hat{\alpha}(k-i)}) \tag{9.1}$$

where $\hat{l}(k)$ is the number of lost packets measured in the k^{th} time interval and $\hat{\alpha}(k)$ is the number of total input packets measured in the same time interval. Both of them are retrieved from the interface counters in standard MIB II. R is the size of the moving window for calculating the packet loss ratio. To accurately calculate $plr(k)$, the R should be large enough. The calculation of packet loss ratio is shown in Figure 9.4.

Based on the basic b-asymptote estimator in Chapter 4, a reactive estimator $re(k)$ for k^{th} time interval is constructed in (5.6).

$$re(k) = lbe(k) + di(k) \tag{9.2}$$

where $lbe(k)$ is calculated with (9.3) and $di(k)$ is a dynamic term added to adjust the error.

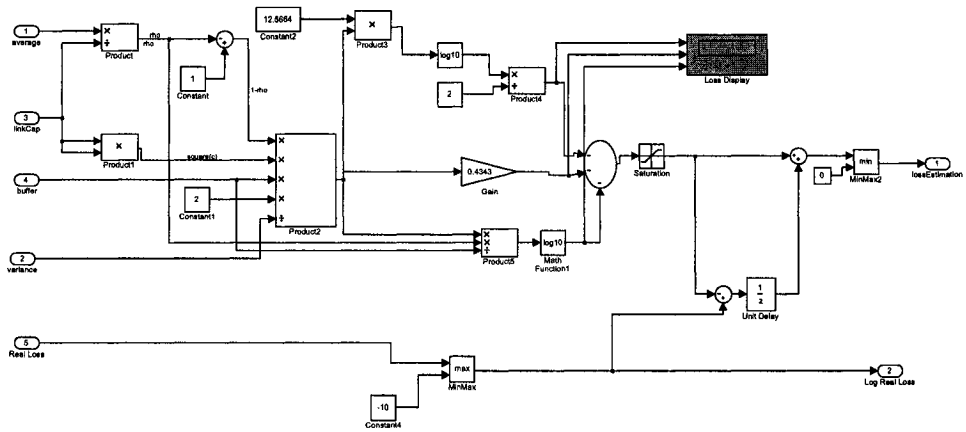


Figure 9.5: Simulation Diagram of the Packet Loss Estimator in Simulink

$$lbe(k) = -\frac{2b(c-\hat{\mu}(k))}{\hat{\sigma}^2(k)}log(e) - log(\frac{2\hat{\mu}(k)(1-\hat{\mu}(k))}{\hat{\sigma}^2(k)}) \tag{9.3}$$

where $\hat{\mu}(k)$ and $\hat{\sigma}(k)$ are estimated traffic mean and variance for the same k^{th} time interval, respectively. They can be calculated as the following:

$$\hat{\mu}(k) = \frac{1}{N}\sum_{i=0}^{N-1}\hat{\alpha}(k-i) \tag{9.4}$$

$$\hat{\sigma}^2(k) = \frac{1}{N-1}\sum_{i=0}^{N-1}[\hat{\alpha}(k-i)-\hat{\mu}(k)]^2 \tag{9.5}$$

where N is the size of the moving window for calculating the average of the traffic mean and variance, and $\hat{\alpha}(k)$ is the measured traffic in the k^{th} time interval.

The estimator for the packet loss probability is implemented and shown in Figure 9.5, where the input includes packet rates mean and variance, link bandwidth and buffer size. The diagram is just the same implementation for the equation (9.3) and it is easily to understand.

A linear prediction technique [77] is used to calculate the dynamic term with the following equation.

$$di(k) = \sum_{l=1}^{p}[plr(k-l) - re(k-l)]w(l) \tag{9.6}$$

where $w(l)s$ are the linear prediction coefficients calculated so that the error due to prediction can be minimized, and p is the size of the moving window.

In this thesis, a simple prediction scheme is used since minimal calculation complexity is required in practical applications. In the analysis and following experimental tests, the moving window average to calculate the predicted dynamic term is employed. If the moving average with window size p is used, the linear prediction coefficients are:

$$w(i) = \frac{1}{p}, \forall i \in [1, p] \tag{9.7}$$

Therefore, the $di(k)$ is simplified into the following equation.

$$di(k) = \frac{1}{p} \sum_{l=1}^{p} (plr(k-l) - re(k-l)) \tag{9.8}$$

Kalman Filter on the Traffic Mean and Variance

Kalman filter is used to provide the best estimate of the mean, μ_k and variance, σ_k, of the instant packet rate of the input traffic process.

The state of the system is defined as $X_k = [\mu_k, \sigma_k]^T$. The dynamics of the system model is defined as:

$$X_k = AX_{k-1} + Bu_k + w_k \tag{9.9}$$

where u_k is the instant packet rate input. The noise, $w_k = [w_k^m, w_k^v]^T$, is white noise source with zero mean and covariance matrix Q and is uncorrelated with the input. A and B are constant matrixes defined as:

$$A = \begin{bmatrix} 1 - 1/N & 0 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1/N & 0 \\ 0 & 0 \end{bmatrix}$$

where N is the sample size.

In the network, the packet rate mean and variance can be measured and expressed by $Z_k = [\bar{\mu}_k, \bar{\sigma}_k]^T$. Therefore, the measurement model is described by

$$Z_k = HX_k + v_k \tag{9.10}$$

where $H = I$ is the constant gain and $v_k = [v_k^m, v_k^v]$ is the measurement noise. The noise, v_k, is also white noise source with zero mean and covariance R that is uncorrelated with

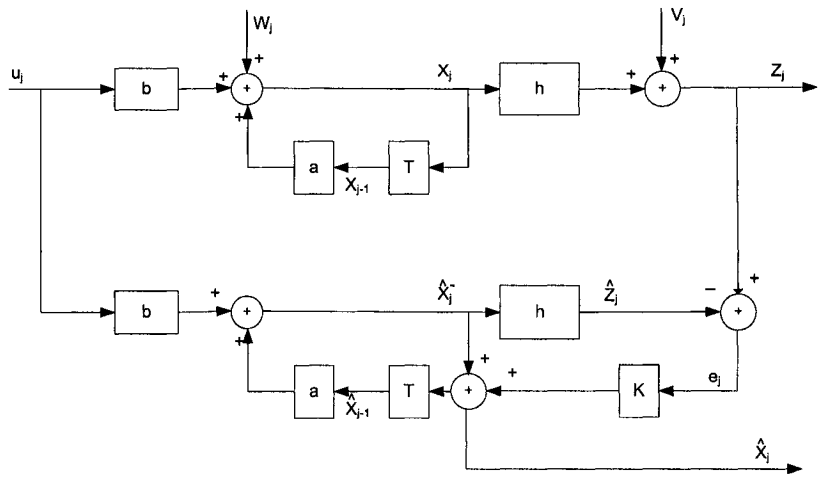


Figure 9.6: System Model and Kalman Filter

the input. The two noise sources are independent of each other and independent of the input.

Given the system depicted above, a Kalman filter is applied to optimally recursive estimate the μ_k and σ_k. The task of the Kalman filter is to filter Z_k so as to estimate the variable X_k while minimizing the effects of w and v.

Kalman filter is a general method for the optimal estimation of a noisy measurement. The basic idea is to use the estimation error obtained from the past measurement of the variables and the estimation equation to fix the one-step prediction. The algorithm runs recursively and is applicable for the on-line application. The system model and Kalman filter approach are shown in Figure 9.6.

The upper part of Figure 9.6 describes the system model, that is specified by equation (9.9) and (9.10). The lower part is used to implement the Kalman filter.

First, the variable, $\hat{X}_k^-$, refers to the *apriori* estimate. It is defined as:

$$\hat{X}_k^- = A\hat{X}_{k-1} + Bu_k \tag{9.11}$$

This *apriori* estimate is used to predict an estimate for the output, $\hat{Z}_k$. The difference between this estimated output and the actual measured output is called residual. The

residual is denoted by e_k and expressed as follows:

$$e_k = Z_k - \hat{Z}_k \tag{9.12}$$

$$= Z_k - H\hat{X}_k^- \tag{9.13}$$

If the residual is small, it generally means a good estimate; if it is large the estimate is not so good. This information is then employed to refine the estimate of X_k. This new estimate is called the *aposteriori* estimate, denoted as $\hat{X}_k$.

$$\hat{X}_k = \hat{X}_k^- + Ke_k \tag{9.14}$$

$$= \hat{X}_k^- + K(Z_k - h\hat{X}_k^-) \tag{9.15}$$

where K is called Kalman gain and the calculation of K is the heart of Kalman filtering.

In order to solve K, two mean squared errors are defined. One is called *apriori* covariance, P_k^-, described as:

$$P_j^- = E(X_k - \hat{X}_k^-)^2 \tag{9.16}$$

The other one is called *aposteriori* covariance and defined as:

$$P_j = E(X_k - \hat{X}_k)^2 \tag{9.17}$$

After calculation, the two covariances are simplified as:

$$P_k^- = AP_{k-1}A^T + Q \tag{9.18}$$

$$P_k = (I - K_k H)P_k^- \tag{9.19}$$

A Kalman filter minimizes the *aposteriori* variance, P_k, by choosing the suitable value of K. Therefore, K is calculated by:

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R} \tag{9.20}$$

Based on the above equations, the recursive Kalman filter algorithm is depicted in the following steps:

1. Setup the initial values for Q, R, P_0 and $\hat{X}_0$.

2. Project the *apriori* estimate with equation (9.11)

3. Project the error covariance for the *apriori* estimate with equation (9.18).
4. Compute the Kalman gain with equation (9.20)
5. Update the estimate with measurement with equation (9.15)
6. Update the error covariance for the *aposteriori* estimate with equation (9.19)
7. goto step 2.

9.2.2 System Identification

Linear modeling is the most common way to describe a dynamical system. The linear model is more mature than nonlinear ones and it is relatively easier to implement. For most of non-linear cases, they can be translated into linear models by changing variables. Since the model is just a description of some major concerned properties of the system, the model does not need to be a true and accurate description of the system, nor does one have to believe so, as long as it serves its purpose. In the study, the linear state space model is assumed for the dynamic packet loss system.

Initial Model Selection

Based on the mechanisms that govern the packet loss system's behavior, an initial mathematical model is obtained. The packet loss system is assumed as a linear time-invariant model at the beginning of the model analysis.

A linear discrete time-invariant model is usually specified by the impulse response, the spectrum response and the transfer function. However, the state-space model is a more common representation of the dynamic model, where it is written as a system of first-order difference equations using an auxiliary state vector $x(k)$. Since in the transfer function the order is 2, the state-space model of the packet loss system is then simplified as a 2-order IIR system:

$$x(k+1) = Ax(k) + Br(k) \tag{9.21}$$

$$L(k) = Cx(k) + Dr(k) + e(k) \tag{9.22}$$

where $x(k)$ is the vector of states, $L(k)$ is the output, $r(k)$ is the input and $e(k)$ is the measurement noise in the k^{th} time interval.

To identify the state-space model of the dynamic system, most of effort involved relates to defining and manipulating the structure. That is how to determinate the structure and values of matrix A, B, C and D.

The effect of the input on the output will be delayed at least one sample and all passes via $x(k)$, so D can be assumed to be zero in order that there is no direct influence from $r(k)$ to $L(k)$.

Because there is no unique state-variable formulation for a given system, the canonical form of the state-variable model is selected in the paper to simplify the processing. The initial model structure can be described as the following four matrixes in the canonical form:

$$A = \begin{bmatrix} 0 & 1 \\ \alpha_0 & \alpha_1 \end{bmatrix} B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$C \begin{bmatrix} \beta_0 & \beta_1 \end{bmatrix} D \begin{bmatrix} 0 & 0 \end{bmatrix}$$

There are four parameters, α_0, α_1, β_0 and β_1 should be identified in the experiments.

Parameter Estimation

After the model structure is selected, the set of candidate models $M(\theta)$ is defined by a parameter vector θ denoted with the following equation.

$$\theta = (\alpha_0, \alpha_1, \beta_0, \beta_1)^T \tag{9.23}$$

The search for the best model within the set then becomes a problem of determining or estimating the proper parameter θ^*. The proper parameter θ^* should be selected in order to satisfy that $M(\theta^*)$ is the best model in all of the model set.

Since the essence of a model is its prediction aspect, the model performance is judged in this respect. Prediction Error Method (PEM) is considered as a kind of generalized framework for system identification. With PEM, model parameters will be estimated by minimizing the optimally determined one-step-ahead output prediction error.

Given the particular parameters θ^* and the measured data set Z,

$$Z = \{z_k | z_k = [L(k), r(k)]\}, k \in [1..N]$$

the best one-step ahead output prediction at k is:

$$\hat{L}(k|\theta^*) = (C[I - A]^{-1}B + D)r(k-1) \tag{9.24}$$

Then, the one-step ahead output prediction error at k, denoted by $\epsilon(k, \theta^*)$, is given by

$$\epsilon(k, \theta^*) = L(k) - \hat{L}(k|\theta^*) \tag{9.25}$$

The PEM minimizes the prediction errors defined by a scalar-valued quadratic function $\frac{1}{2}\epsilon^2$. Then the parameter estimation is to search the θ^* to minimize the following value $V(\theta)$:

$$V(\theta) = \frac{1}{2N}\sum_{k=1}^{N}(\epsilon(k, \theta))^2 \tag{9.26}$$

In general, (9.26) cannot be minimized by analytical methods. The solution has to be found by iterative, numerical techniques. From the initial estimate, a Gauss-Newton minimization procedure [75] is carried out until the norm of the Gauss-Newton direction is less than a certain tolerance. With Gauss-Newton algorithm, the iterative searching routine updates the estimate of the minimizing point according to

$$\hat{\theta}^{i+1} = \hat{\theta}^i - [H(\hat{\theta}^i)]^{-1}V'(\theta^i) \tag{9.27}$$

where $\hat{\theta}^i$ denotes the i^{th} iterate for the estimate of θ and $V'(\theta)$ denotes the gradient of $V(\theta)$, and $H(\theta)$ is the Hessian matrix that modifies the search direction. The selection of $H(\theta)$ is obtained using the numerical procedure of the descent algorithm and guarantees convergence to a stationary point. It is calculated by the following equation:

$$H(\theta) = \frac{1}{N}\sum_{k=1}^{N}\psi(k, \theta)\psi^T(k, \theta) \tag{9.28}$$

where $\psi(k, \theta)$ denotes the gradient of the error. That is,

$$\psi(k, \theta) = \epsilon'(k, \theta)$$

The major burden lies in the calculation of the sequence of prediction errors and their gradients. The calculation of $H(\theta)$ and of the θ is done by using *pem* in this thesis.

Recursive Prediction Error Method

The time-variability of the response function and the variable parameters suggest that an adaptive system identification technique might be useful. Not only would it be able to systematically vary model parameters to track non-stationary dynamics in time, but the recursive nature of most adaptive algorithms is ideal for implementation in an on-line operational space model, since the state and parameter estimates can be updated without repeated computation of matrix solution which can be quite time consuming. Furthermore, the old data may be discarded and potential data storage problems are also bypassed. In the next experiment, the on-line data is captured and the parameters are identified by the general Recursive Prediction Error Method (RPEM).

The main advantage of the RPEM is its general applicability. The algorithm can be specified in different cases as special names. The RPEM estimates the parameters by minimizing the optimally determined one-step-ahead output prediction error denoted by $\epsilon(k)$. Since information in the latest pair of measurements ($L(k)$, $r(k)$) normally is small compared to the information already accumulated from previous measurements, the algorithm typically utilizes the previous estimated parameters to ensure that the parameters can be calculated during a sampling interval. Based on such an idea, the algorithm estimates the parameters in a recursive way without repeated computation of matrix solutions that can be quite time consuming.

Analogous to the PEM case, let us consider a weighted quadratic prediction-error criterion:

$$V(\theta) = \gamma(N)\frac{1}{2}\sum_{k=1}^{N}\beta(k)\epsilon^2(k) \tag{9.29}$$

where $\beta(k)$ is the weight. $\gamma(N)$ is used to normalize the gain and defined as the following:

$$\gamma(N) = [\sum_{k=1}^{N}\beta(k)]^{-1} \tag{9.30}$$

Applying the recursive Gauss-Newton prediction-error algorithm [75], the recursive scheme is given by the following group of equations:

$$\epsilon(k) = L(k) - \hat{L}(k) \tag{9.31}$$

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \gamma(k)R^{-1}(k)\psi(k)\epsilon(k) \tag{9.32}$$

$$R(k) = R(k-1) + \gamma(k)[\psi(k)\psi^T(k) - R(k-1)] \tag{9.33}$$

where $R(k)$ is the second derivative of $V(\theta)$ with respect to θ termed the Hessian. $\psi(k)$ is the gradient of the predictions. $\hat{\theta}(k)$ is the estimate of the parameter vector. All of the variables are termed for the k^{th} time interval.

9.2.3 Robust Controller Design

The identified linear model contains some uncertain parameters because of the linearization process. The controller should be designed to perform satisfactorily over the anticipated range of plant parameter variations.

However, robustness trades with performance. In this thesis, a robust controller is designed with best performance for specified plant-parameter robustness. A Linear Matrix Inequality (LMI) approach for designing the Proportional-Integral (PI) controller for the identified dynamic system is studied.

Linear Matrix Inequality

A Linear Matrix Inequality (LMI) [18] is any constraint of the form

$$A(x) := A_0 + \sum_{i=1}^{n} x_i A_i < 0 \tag{9.34}$$

where $x \in R^n$ is the variable vector to be solved and the symmetric matrices $A_i \in R^{n \times n}$, i = 0,...,n are given constant matrices. The inequality (< 0) stands for negative definite, i.e., the largest eigenvalue of $A(x)$ is negative.

The importance of the LMI arises in system and control theory because the controller design can be formulated as convex optimization problems with LMIs that are amenable to computer solution.

As a result, finding a solution x^* to (9.34), is a convex optimization problem. Convexity has an important consequence: even though (9.34) has no analytical solution in general, it can be solved numerically with guarantees of finding a solution when one exists, which has great practical importance.

The canonical expression in (9.34) is generic. However, LMIs rarely arise with this format in control applications, since this expression is less intuitive and transparent. Moreover, the number of matrices grows roughly as $\frac{n^2}{2}$ if n is the size of the A matrix, so that the canonical form is very inefficient from a storage viewpoint. According to the reasons above, the LMI Lab uses a structured representation of LMIs.

In general, LMIs assume a block matrix form where each block is an affine combination of the matrix variables.

$$L(X,\gamma) = \begin{bmatrix} A^TX + XA & XC^T & B \\ CX & -\gamma I & D \\ B^T & D^T & -\gamma I \end{bmatrix} < 0 \tag{9.35}$$

where $X \in R^{n\times n}$ and $\gamma \in R$ are the matrix variables of the problem. A, B, C, D are given matrices. $L(X,\gamma)$ is a symmetric block matrix. Each block of $L(X,\gamma)$ is an affine expression in the matrix variables X and γ.

In order to solves the LMI problems, the LMI LAB within the Control Toolbox [40], which implements state-of-the-art interior-point LMI solvers, is employed in this thesis. The interior-point methods by Nesterov [82] are extremely efficient and are significantly faster than classical convex optimization algorithms. Their high performance is achieved through C Matlab Executable (C-MEX) implementation and by taking advantage of the particular structure of each LMI. C-MEX files are functions written in C, which can run on any system that is already running MATLAB.

Model Uncertainty and Robust Analysis

The term *uncertainty* refers to the differences or errors between models and reality [134]. To be practical, the identified system model cannot exactly describe the physical system. The inevitable inadequacy exists in all of the models, because the model is highly simplified; otherwise it would be too complicate to handle for a controller design procedure. Therefore, the controller design is based on a crude mathematical model of the physical reality.

There are two basic types of uncertainty: parameter uncertainty and unstructured uncertainty. In this chapter, only the parameter uncertainty is dealt with, since the identified model only has uncertain parameters, which are expressed as:

$$A = A_0 + \delta A$$

$$C = C_0 + \delta C$$

where A_0 and C_0 are known constant matrices names the nominal parameters. δA and

δC are matrices of uncertainties and considered to be of the affine type:

$$\delta A = \sum_{j=1}^{p} \epsilon_j A_j$$

$$\delta C = \sum_{j=1}^{p} \epsilon_j C_j$$

where $|\epsilon_j| < 1$ are unknown parameters; A_j and C_j are constant matrices. In such a way, δA and δC belong to a polytopic set:

$$\Omega = Cov\{[A_i,\ C_i] |\ i \in [1..m]\} \tag{9.36}$$

where Cov refers to a convex hull, m is the number of the polytope vertices.

Generally speaking, the solution for the LMIs must be searched all over the uncertain system Ω. Due to the properties of the polytopic system, solutions can be sought only at the polytopic vertices instead of all points within the polytope. Therefore, a solution to LMIs only needs to be sought at the vertices $[A_i, C_i]$, where $i \in [1..m]$ and hence the task for solving the constraints is much reduced.

The stability of the uncertain model is whether all trajectories of the system converge to 0 as $k \rightarrow \infty$. A necessary and sufficient condition [18] for this is that there exists a variable matrix P to satisfy the following inequalities:

$$P > 0,\ A_i^T P + P A_i + C_i^T C_i < 0, i = 1, ..., m \tag{9.37}$$

Linear Quadratic Optimal Control

The optimal design procedure minimizes the cost function and therefore yields a best control system. Consider the linear dynamic system (9.22), the cost function has the form:

$$J_N = \sum_{k=0}^{\infty} L[l(k),\ r(k),\ u(k)] \tag{9.38}$$

Usually a quadratic cost function of the states and the control signals is studied in the optimal control literature, since the development is simple and the cost function is logical in practice. This technique is named Linear Quadratic Regulator (LQR).

The standard LQR problem is to determine the control law u of the form:

$$u(k) = -Kx(k) \tag{9.39}$$

which minimizes the quadratic cost:

$$J(u) = \sum_{0}^{\infty}(x^T Qx + u^T Ru) \tag{9.40}$$

for any initial state x(0), where Q and R are symmetric positive semi definite matrix and positive definite matrix, respectively. It turns out that the solution u^* to this optimal control problem can be expressed in the state feedback form:

$$u^* = -Kx = -R^{-1}B^T Px \tag{9.41}$$

where P is the symmetric positive definite solution of the Algebraic Riccati Equation (ARE):

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{9.42}$$

and the minimum quadratic cost is given by

$$J_{min} = x^T(0)Px(0) \tag{9.43}$$

The solution to the LQR problem relies on solving the algebraic Riccati equation (9.42). An efficient alternative for this problem is the LMI technique due to this convexity. By the LMI technique, the LQR problem can be rephrased as an optimization problem over P and Y:

$$Minimize\ x^T(0)Px(0) \tag{9.44}$$

subject to

$$\begin{bmatrix} AP^{-1} + P^{-1}A^T + BY + Y^T B^T & P^{-1} & Y^T \\ P^{-1} & -Q^{-1} & 0 \\ Y & 0 & -R^{-1} \end{bmatrix} \leq 0 \tag{9.45}$$

where $P^{-1} > 0$ and $Y = R^{-1}B^T$.

In the practice situation, the objective (9.44) is modified as

$$x^T(0)Px(0) \leq \gamma \tag{9.46}$$

where γ is the specified upper bound. This inequality is also expressed as an LMI:

$$\begin{bmatrix} \gamma & x^T(0) \\ x(0) & P^{-1} \end{bmatrix} \geq 0 \tag{9.47}$$

Consequently, the optimization problem is converted to seek a solution (P^*, Y^*) that satisfies both of LMIs in (9.45) and (9.47). Then, the state feedback gain is given by:

$$K = -Y^* P^* \tag{9.48}$$

Robust Controller Design

The advantage of using LMI is its convenience to include other specifications for the controller design, since the controller is usually required to meet various specifications simultaneously in practical situations. Various design specifications are recast into the LMIs and the resulting LMI constraints are solved in polynomial time by using recently developed convex optimization algorithms.

In this paper, a PI controller is designed in the state-space setting for the ease of using LQR-LMI approach. A PI controller has the following transfer function:

$$D(z) = K_p + K_i \frac{z+1}{z-1} \tag{9.49}$$

In the state-space model, the PI controller design becomes a static state feedback controller, and the static feedback gain K simply contains all the PI controller parameters.

For the design specification, it is well known that the transient response of the dynamic system is closely related to the locations of poles. The desired damping can be achieved by constraining the closed loop pole to lie in a prescribed region. The conditions for the pole placement constrain is also recast in the LMIs and LMI region.

Let $L = L^T = [\lambda_{i,j}]_{1 \le i,j \le m}$ and $M = M^T = [\mu_{i,j}]_{1 \le i,j \le m}$, which means that L and M are mxm matrix with the elements λ_{ij} and $\mu_{i,j}$, respectively. The closed-loop poles lies in the LMI regions:

$$Z \in \mathcal{C} : L + Mz + m^T \bar{Z} < 0 \tag{9.50}$$

if and only if there exists a symmetric positive definite matrix $\hat{P}$ satisfying the following LMI[25]:

$$[\lambda_{jk} + \mu_{jk}(A\hat{P} + BY) + \mu_{kj}(A\hat{P} + BY)^T]_{1 \le j,k \le m} < 0 \tag{9.51}$$

The robust PI controller design procedure is shown in the following steps:

1. A set of multiple models are derived from the identified uncertain model according to (9.36)
2. Set the LQR weighting matrix Q and R for (9.40)
3. Find the solution (P^*, Y^*) to LMIs (9.45), (9.47) and (9.51)
4. Calculate the PI controller parameters from (9.48)

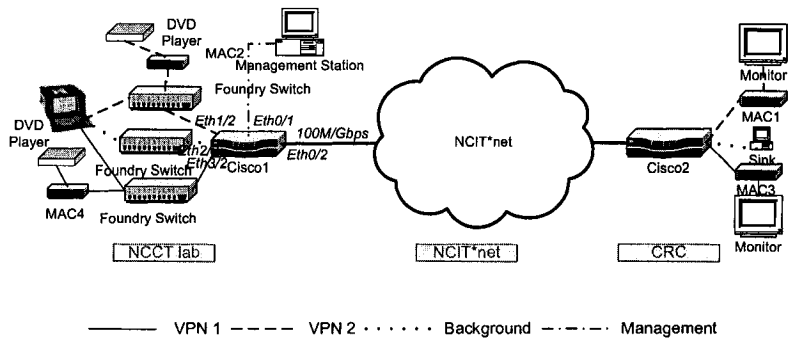


Figure 9.7: Test Bed in NCIT*net2 network

9.3 Results Obtained on the Packet Loss Control Loop

This section presents the design of the network testbed and the measurement methodology used for data collection. All of the traffic are measured from the live network and stored in different data files. A Matlab simulation program is designed to read the real traffic data and apply the designed controller to filter the signals.

9.3.1 Traffic Measurement

All of the experiments take place across the National Capital Institute of Telecommunication's NCIT*net 2 network [39]. NCIT*net 2 is an optical research network that supports both live and experimental traffic and therefore is looked as a live networking laboratory that supports research in data communication at any layer of the OSI model.

100M Testbed Design

Figure 9.7 displays the network topology for measuring 100M packet rates used in experimental study. Three BGP/MPLS VPN services are set as services between the two Cisco routers (192.168.2.1 and 192.168.3.1, respectively), where the connection bandwidth is set to 100M/1Gbps. The first VPN is composed by one DVD flow from MAC4 to MAC3 and one background Ethernet traffic from the GN Nettest to MAC3. The second VPN also has a DVD MPEG2 video flow from MAC2 to MAC1 and an Ethernet background traffic flow from GN Nettest to MAC1. VPN3 generates the background

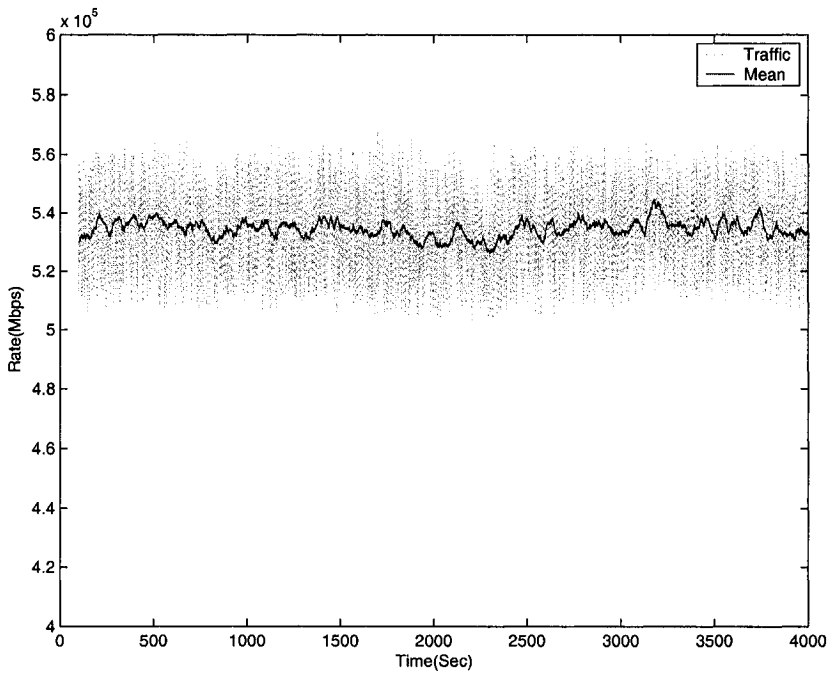


Figure 9.8: Traffic Measurement on the Eth1/2 of Cisco1 in NCCT Network

traffic, which can simulate many on-off Ethernet traffic flows. All of the VPN services are aggregated on the core link between the two Cisco routers. One management workstation is connected with all of the devices through Interface eth0/1 in the first Cisco router to measure and control the VPN services and traffic. It is used for the definition, creation and activation of the VPN services above, and for all adjacent network management and element management operations. The management traffic is separated from the experimental VPN data traffic through configuring different VLANs in the network.

The following traffic in Figure 9.8 is one sample measured from this testbed.

Gigabit Testbed Design

Figure 9.9 shows the NCCT lab and NCIT*net2 network, while Figure 9.10 shows the network topology designed for the experimental study. An OC-192 port of the traffic generator G sends multiple traffic streams (IP packets having uniformly-distributed destination addresses within a range) over a point-to-point link to the router designated R1. R1 sends each packet out one of the three Gigabit Ethernet interfaces to R2, in a round-robin manner; this emulates the behavior of large-scale IP networks, which do not provide any guarantees as to which path a packet will take when there are multiple

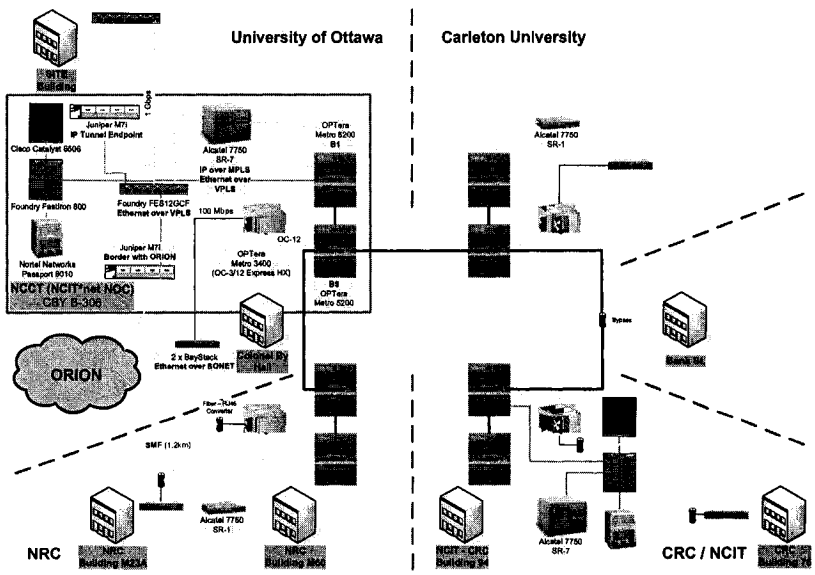


Figure 9.9: NCCT lab and NCIT*net2 Network

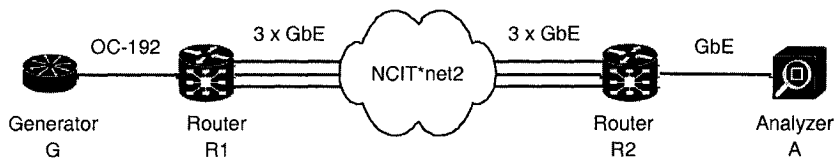


Figure 9.10: Test Bed across NCIT*net2

paths to a certain destination. R2 forwards the three traffic sources out the same Gigabit Ethernet link to the analyzer A.

The analyzer provides per-flow or aggregated statistics. The output provided is of the form (r, l), where r is the number of packets received per sampling interval, and l is the number of packets lost per sampling interval. The sampling interval is $\Delta t = 0.1$ seconds. The packet loss ratio (plr) for each sampling interval can be expressed as

$$plr = \frac{l}{r + l} \tag{9.52}$$

and the packet loss rate is l packets per sampling interval or $\frac{l}{\Delta t}$ packets per second.

It is obvious that no packet loss will occur on the interface between R2 and A if the generator rate R_G is less than 1 Gbps. As soon as R_G exceeds 1 Gbps, packet loss starts to be observed on the interface. The maximum R_G acceptable for this topology

is 3 Gbps since loss will occur on the egress of the three R1 interfaces for higher values as well.

Therefore, a number of data collecting sessions have been run for different aggregate traffic volumes, starting at 1.25 Gbps and in increments of 0.25 Gbps. Each session spans over at least 100000 sampling intervals (10000 seconds or 2 hours and 50 minutes, approximately). It ensures that enough data is collected so that system identification is as accurate as possible.

The following traffic and packet loss ratio is measured from the Gigabit Ethernet link.

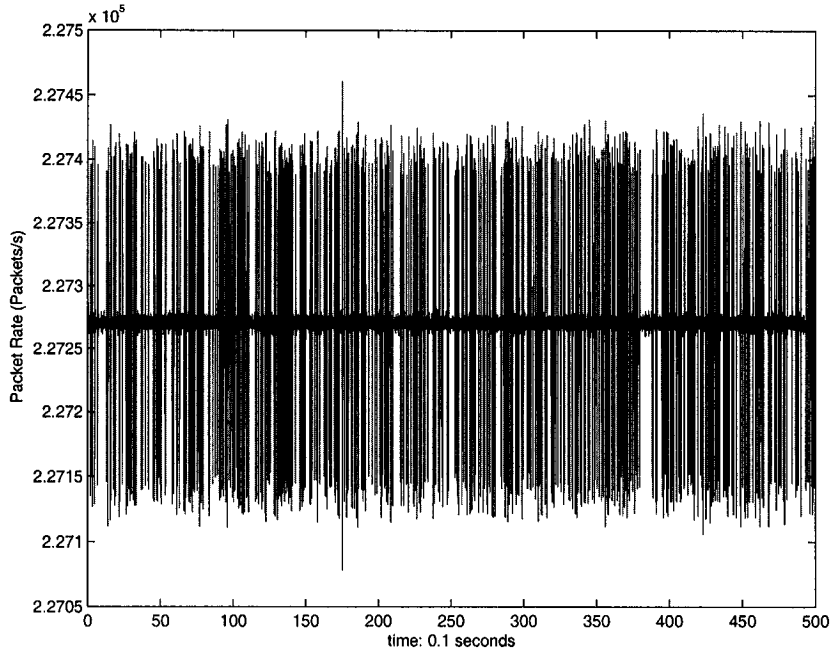


Figure 9.11: Packet Rates Measured in the Gigabit Ethernet Link between R1 and R2

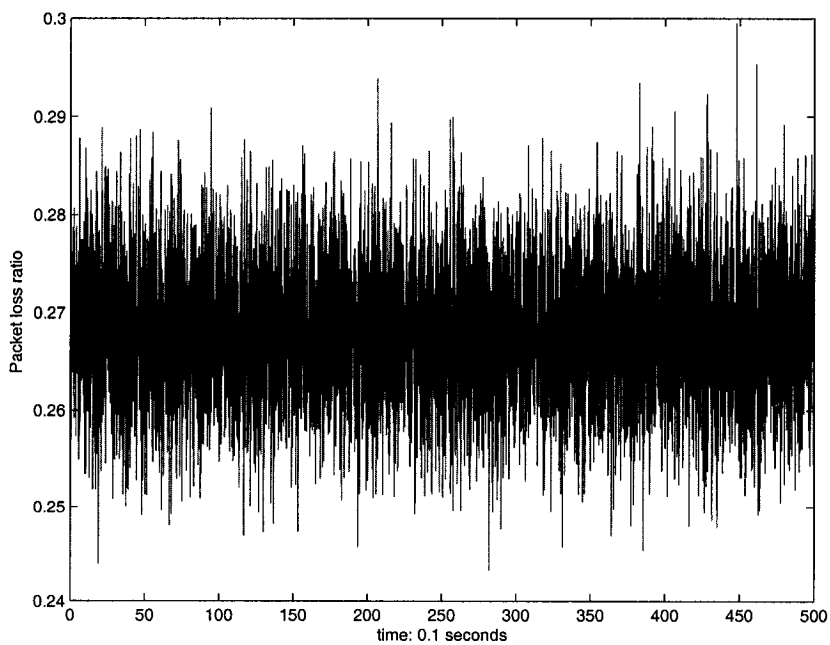


Figure 9.12: Packet Loss Ratio Measured in the Gigabit Ethernet Link between R1 and R2

9.3.2 Loop Parameter Calculation and Results Obtained

Based on the measured traffic data from 100Mbps link shown in Figure 9.8, the system parameters are identified as follows:

The nominal matrix:

$$A = \begin{bmatrix} 0 & 1 \\ -0.323 & 0.618 \end{bmatrix} \quad C = \begin{bmatrix} 0.236 & 0.542 \end{bmatrix}$$

The interval values for the four uncertain parameters are:

$$\alpha_0 \in [0.476\ 0.721], \quad \alpha_1 \in [-0.383\ -0.244]$$
$$\beta_0 \in [0.512\ \ 0.594], \quad \beta_1 \in [0.251\ \ 0.338]$$

The vertices of the corresponding polytopic model are:

$$A_1 = \begin{bmatrix} 0 & 1 \\ -0.383 & 0.476 \end{bmatrix} \quad C_1 = \begin{bmatrix} 0.251 & 0.512 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & 1 \\ -0.383 & 0.721 \end{bmatrix} \quad C_2 = \begin{bmatrix} 0.251 & 0.594 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0 & 1 \\ -0.244 & 0.476 \end{bmatrix} \quad C_3 = \begin{bmatrix} 0.338 & 0.512 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} 0 & 1 \\ -0.244 & 0.721 \end{bmatrix} \quad C_4 = \begin{bmatrix} 0.338 & 0.594 \end{bmatrix}$$

The online identification of the four parameters is shown in Figure 9.13. And the variance of the parameters is shown in Figure 9.14.

Specify the LQR cost weight as $Q = 1$, $R = 1$, $\gamma = 200$. By the proposed method, the PI controller parameters are obtained as:

$$K = [K_p,\ K_i] = [0.301,\ 0.505] \tag{9.53}$$

The error is shown in Figure 9.15, and the closed loop response of the system is shown in Figure 9.16. Although error is variable, the value is within a certain interval.

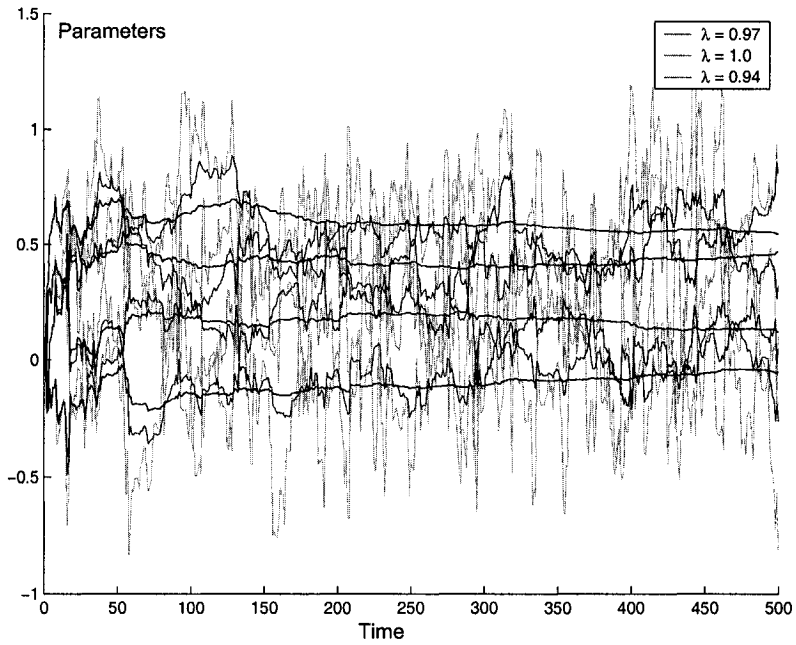


Figure 9.13: On-line Identification of the Parameters Based on Traffic Measured from the Eth1/2 of Cisco1 in NCCT Network

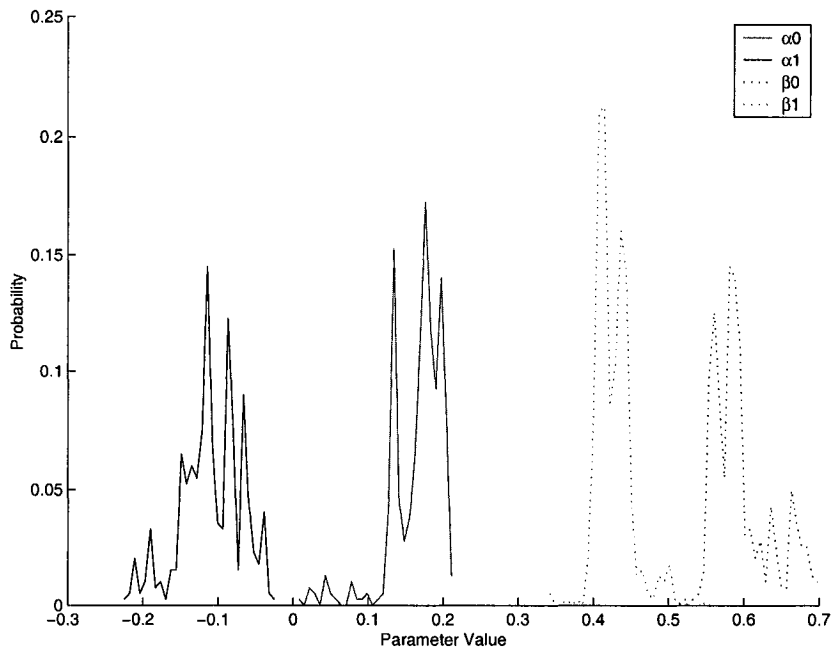


Figure 9.14: Distribution of the Parameter Values Based on Traffic Measured from the Eth1/2 of Cisco1 in NCCT Network

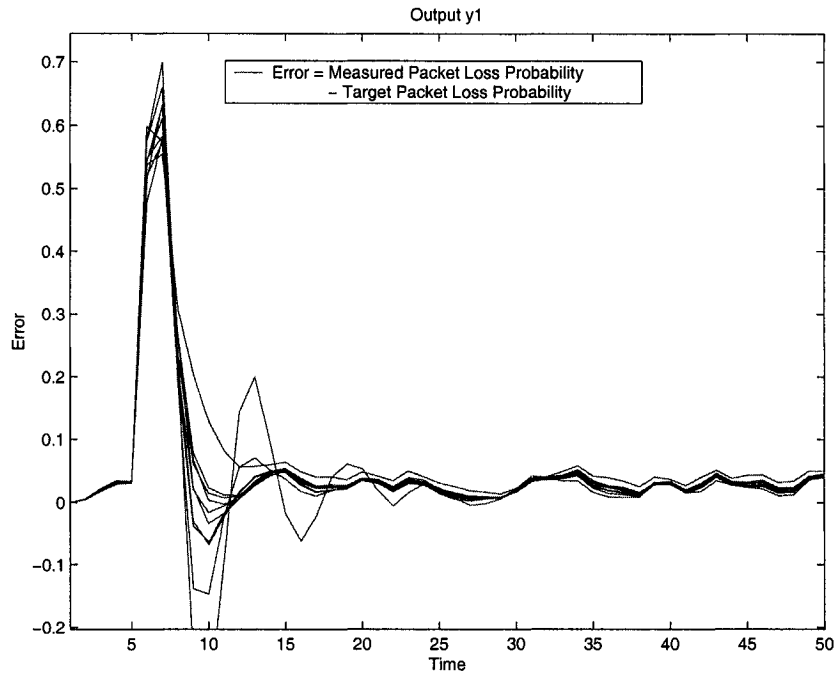


Figure 9.15: Closed Loop Response for the Traffic Measured from the Eth1/2 of Cisco1 in NCCT Network

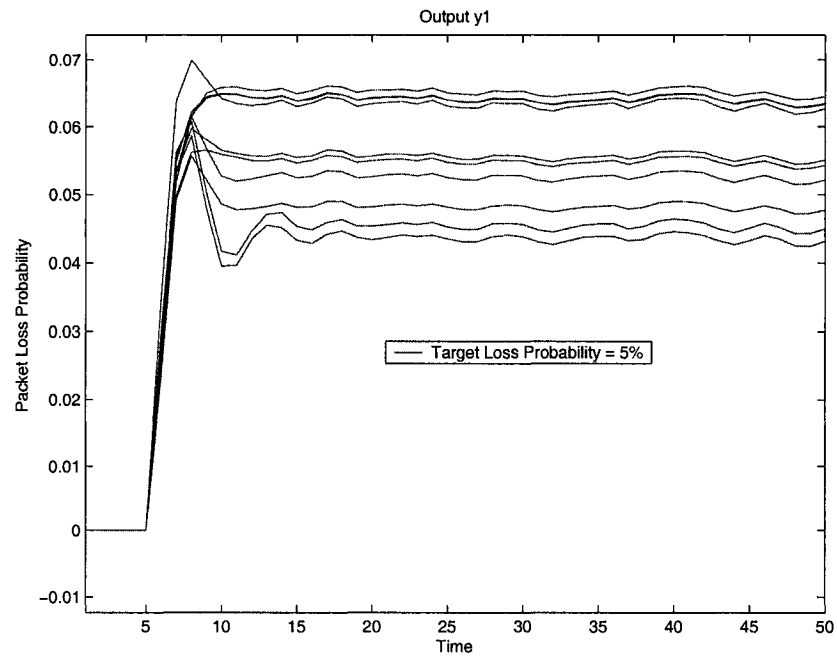


Figure 9.16: Packet Loss Dynamics for the Traffic Measured from the Eth1/2 of Cisco1 in NCCT Network when Target Loss Probability is Set to 5%

In the second example, based on the measured Gbps traffic data from G Ethernet link shown in Figure 9.11, the identified system model parameters are shown as follows:

The nominal matrix:

$$A = \begin{bmatrix} 0 & 1 \\ -0.362 & 0.637 \end{bmatrix} \quad C = \begin{bmatrix} 0.206 & 0.515 \end{bmatrix}$$

The interval values for the four uncertain parameters are:

$$\alpha_0 \in [0.498\ 0.713], \quad \alpha_1 \in [-0.373\ -0.334]$$
$$\beta_0 \in [0.482\ \ 0.553], \quad \beta_1 \in [0.191\ \ 0.313]$$

The vertices of the corresponding polytopic model are:

$$A_1 = \begin{bmatrix} 0 & 1 \\ -0.373 & 0.498 \end{bmatrix} \quad C_1 = \begin{bmatrix} 0.191 & 0.482 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & 1 \\ -0.373 & 0.713 \end{bmatrix} \quad C_2 = \begin{bmatrix} 0.191 & 0.553 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0 & 1 \\ -0.334 & 0.498 \end{bmatrix} \quad C_3 = \begin{bmatrix} 0.313 & 0.482 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} 0 & 1 \\ -0.334 & 0.713 \end{bmatrix} \quad C_4 = \begin{bmatrix} 0.313 & 0.553 \end{bmatrix}$$

The online identification of the four parameters is shown in Figure 9.17. And the variance of the parameters is shown in Figure 9.18.

Specify the LQR cost weight as $Q = 1$, $R = 1$, $\gamma = 200$. By the proposed method, the PI controller parameters are obtained as:

$$K = [K_p,\ K_i] = [0.231,\ 0.382] \tag{9.54}$$

The error is shown in Figure 9.19, and the closed loop response of the system is shown in Figure 9.20. Error is variable but within a certain interval.

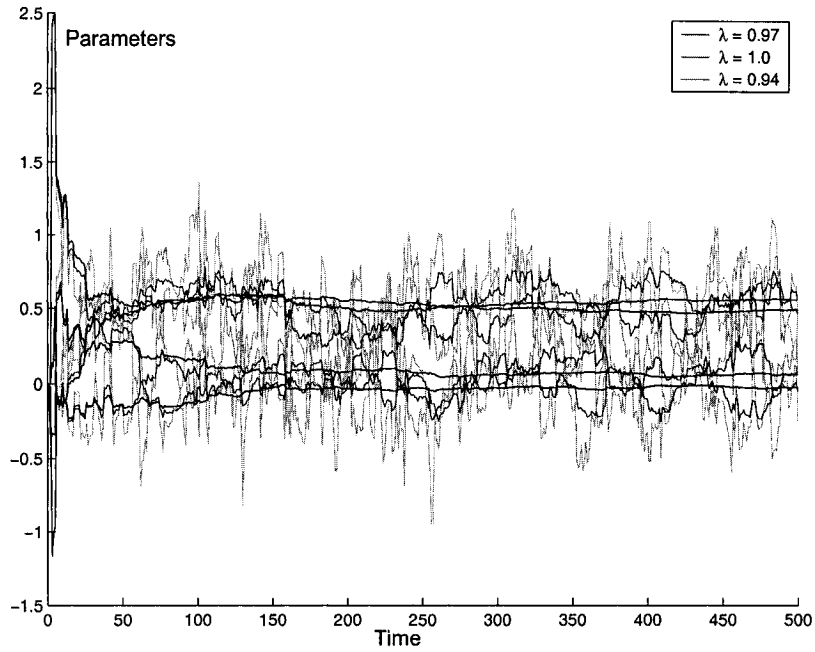


Figure 9.17: On-line Identification of the Parameters Based on Traffic Measured from the G-Ethernet Link in NCCT Network

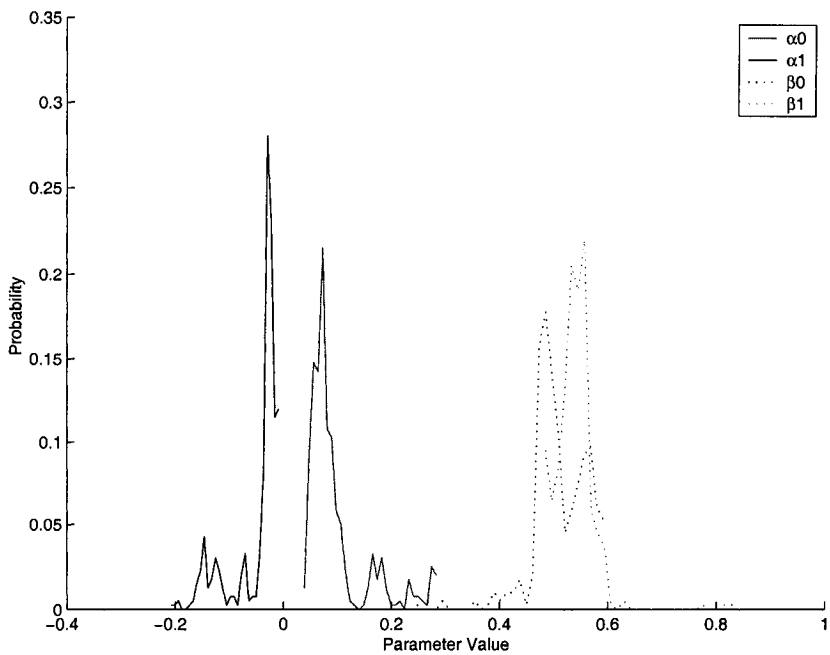


Figure 9.18: Distribution of the Parameter Values Based on Traffic Measured from the G-Ethernet Link in NCCT Network

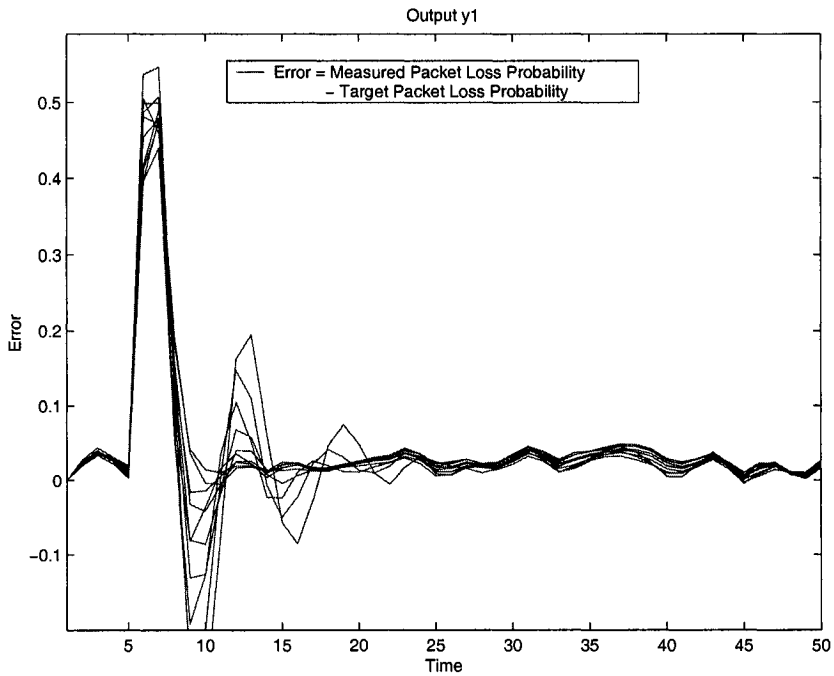


Figure 9.19: Closed Loop Response for the Traffic Measured from the G-Ethernet Link in NCCT Network

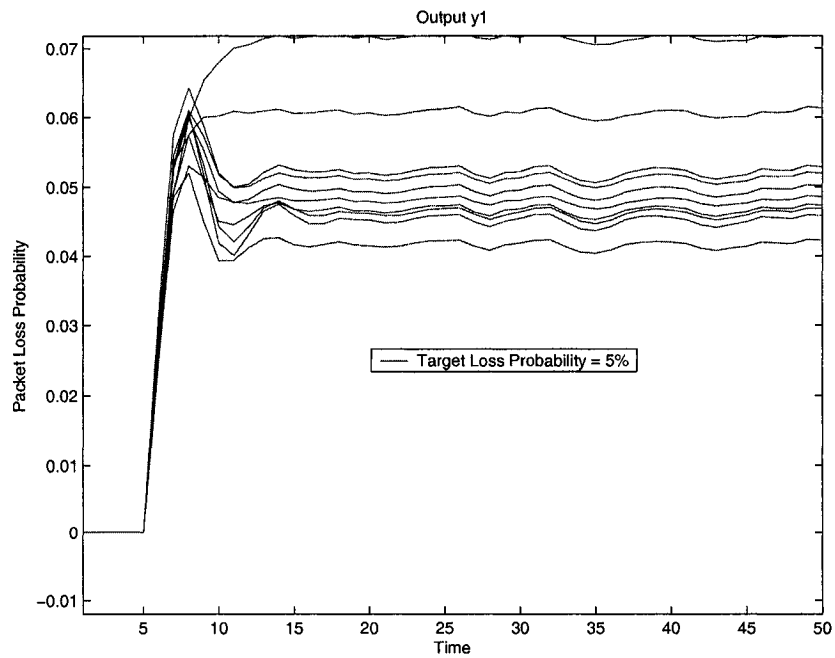


Figure 9.20: Packet Loss Dynamics Based on the Traffic Measured from the G-Ethernet Link in NCCT Network when Target Loss Probability is Set to 5%

9.4 Conclusion

This chapter presents the dynamic QoS control framework for VPN services in the provider's backbone network. The dynamic packet loss control system maintains the preset packet loss probability around a certain value and tries to provide long term statistical QoS for customers. The experimental results show that the proposed controller maintains the loss value within a certain range of the preset target.

Chapter 10

Conclusion and Future Work

The final chapter will first briefly summarize the contributions made in both theoretical results and their applications. The suggestions are then given for further research.

10.1 Contribution of the Thesis

Providing virtual private network services over the IP/MPLS network represents an important addition to a network service provider's revenue, while QoS control and guarantees is one of the key issues to be addressed for efficiently provisioning such services.

Some network service providers have over provisioned their network resources in order to guarantee bandwidth availability, but this is usually an expensive solution. Furthermore, the over-provisioning does not prevent the temporal demand fluctuation or the network failure. To win in the competition, providers should better utilize their resources, provide more than best-effort service to its customer's application and make profits in the process.

This thesis proposes a QoS control system for the MPLS VPN services, where packet loss probability is the primary QoS parameter and controlled below a preset level. The main works done in the thesis are concluded as follows:

1. As a feedback parameter used in the control system, the packet loss probability is first estimated with two methods [118, 127].

2. A reactive estimator is proposed to estimate the packet loss probability on line [117, 129], where Kalman filter is employed to optimally estimate the packet rate mean and variance [125].

3. Based on the real-time estimation, the VPN service admission control algorithm is presented [120, 114].

4. With the estimation of packet loss probability as a transducer, a dynamic packet loss control scheme is studied [121, 115], where the model is first identified [122, 123, 116] and the controller is designed with both input-output method and modern control technologies [121, 115].

5. The robust and optimal controller is investigated for the identified models with uncertain parameters [122, 123, 116], as the identification of the plant model lead to considering random variability of the identified system.

In summary, a QoS parameter control scheme is proposed in the thesis for provisioning MPLS VPN services [119]. With the QoS enabled MPLS VPN services, service providers will support enterprise-subscriber voice, video, and data network service offerings.

10.2 Suggestion for Further Research

This thesis proposes a control-theory based QoS control system for the MPLS VPN services. There are still many other interesting directions and approaches to be explored. Some suggestions for the further research work are discussed below.

1. Utilize dynamic programming to maximize the profits. When congestion happens, input rates of specific sources should be throttled. It is an interesting topic how to maximize total profits by decreasing the rates of selected inputs.

2. Combine traffic engineering with MPLS VPN service. The MPLS VPN connections are presumed fixed in this thesis until now. How to optimize the network and select the best path for the VPN services is another valuable research topic. This issue is generally known as Traffic Engineering (TE) in the field of telecommunication management system.

3. Apply the proposed approach in this thesis for the differentiated services. As various types of network traffic such as multimedia traffic are emerging in the Internet, certain different treatments for them gains attention and involves many different processes. Currently, the DiffServ architecture is being developed for

cases of different cost requirements and different degree of guarantee. Applying the approach proposed in this thesis for the Assured Forwarding (AF) and Expedited Forwarding (EF) differentiated services is also an interesting topic.

4. Provide End-To-End (ETE) QoS. ETE QoS can be divided into three parts: customer premise QoS, access network QoS, and core internet QoS. QoS is defined by the bilateral contract or agreement (SLA: Service Level Agreements) between two interconnected parties. The difficulty of this scheme is the fact that the packets traverse several network domains, which are managed by different service providers and may be using different mechanisms. Achieving end-to-end QoS objectives requires the disaggregation of requirements to subnetworks, which may span multiple service providers. Research should proceed on how a broker would coordinate with service providers to offer predictable subject quality to customers, how it can be scalable with rapid establishment, how pricing and division of revenues would be set properly, and how customers would specify their quality/price preferences.

The suggestions above will enrich the QoS control approach and also make it possible for the real application of the system. These developments are highly anticipated in the future.

Bibliography

[1] H. Akaike. Power spectrum estimation through autoregression model fitting. *Ann. Inst. Stat. Math.*, 21:407–419, 1969.

[2] M. Allman, V. Paxson, and W. Stevens. *TCP Congestion Control.* Internet RFC 2581, April 1999.

[3] G. Almes. *A One-way Delay Metric for IPPM.* Internet RFC 2679, September 1999.

[4] G. Almes. *A Round-Trip Delay Metric for IPPM.* Internet RFC 2681, September 1999.

[5] D. Anick, D. Mitra, and M.M. Sodhi. Stochastic theory of a data handling system with multiple sources. *Bell Sys. Tech Journal*, 61(8), 1982.

[6] Grenville Armitage. *Quality of service in IP Networks.* Macmillan Technical Publishing, 2000.

[7] R.R. Bahadur and R.R. Rao. On deviations of the sample mean. *Ann. Of Mathematical Statistics*, (23):1015–1027, 1960.

[8] D. Bansal and H. Balakrishnan. Binomial Congestion Control Algorithms. In *INFOCOM 2001*, 2001.

[9] Paul Barford and Joel Sommers. Comparing probe-and router-based packet-loss measurement. *IEEE INTERNET COMPUTING*, pages 50–56, 2004.

[10] L. Benmohamed and S. M. Meerkov. Feedback Control of Congestion In Packet Switching Networks: The case of a single congested node. *IEEE/ACM Transactions in Networking*, 1:693–708, 1993.

[11] J.C.R. Bennet and H. Zhang. Hierarchical packet fair queuing algorithms. *IEEE/ACM Transactions on Networking*, 5:675–689, 1997.

[12] A.W. Berger and W. Witt. Effective bandwidths with priorities. *IEEE/ACM Transactions on Networking*, 6(4), 1998.

[13] W. Bergman. Narrowband frame relay congestion control. In *Proceedings of the Tenth Annual Phoenix conference of Computers and communications*, March 1991.

[14] G. Bianchi and all. Throughput Analysis of End-to-End Measurement Based Admission Control. In *INFOCOM*, 2000.

[15] S. Black and all. *An Architecture for Differentiated Services.* IETF RFC 2475, 1998.

[16] F. Blanchini, R.L. Cigno, and R. Tempo. Robust Rate Control for Integrated Service Packet Networks. *IEEE/ACM Transactions in Networking*, 5:644–652, 2002.

[17] S. Boyd. A note on parametric and nonparametric uncertainties in control systems. *Proc. American Control Conf.*, pages 1847–1849, 1986.

[18] S. Boyd, L. EL Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in Systems an Control Theory.* SIAM books, Philadelphia, 1994.

[19] B. Braden. *Recommendations on Queue Management and Congestion Avoidance in the Internet.* Internet RFC 2309, April 1998.

[20] Lawrence S. Brakmo and Larry L. Peterson. TCP vegas: End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, 1995.

[21] Lee Breslau and Sugih Jamin. Comments on the performance of measurement-based admission control algorithms. In *INFOCOM'2000*, 2000.

[22] R. Callon, P. Doolan, N. Feldman, G. Swallow, and A. Viswanathan. A framework for multiprotocol label switching. *IETF Internet Draft*, September 1999. draft-ietf-mpls-framework-05.txt.

[23] C.S. Chang. *Performance Guarantees In Communication Networks.* Springer-Verlag, 2000.

[24] K. Chen, K. Ho, and V. Saksena. Analysis and design of a highly reliable transport architecture for isdn frame relay networks. *IEEE Journal on Selected Areas in Communications*, 7:1231–1242, 1989.

[25] M. Chiliali and P. Gahinet. H design with pole placement constraints: an lmi approach. *IEEE Trans. On Auto. Control*, pages 358–367, 1996.

[26] D. Chiu and R. Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[27] Mikkel Christiansen, Kevin Jeffay, David Ott, and F. Donelson Smith. Tuning red for web traffic. In *ACM SIGCOMM 2000 Conference*, pages 139–150, February 2000. Appears in *IEEE/ACM Transactions on Networking*, 9(3):249–264, June 2001.

[28] The ATM Forum Technical Committee. Traffic Management Specification Version 4.1. Technical report, ATM Forum, http://www.atmforum.org, 1999.

[29] C. Courcoubetis, A. Dimakis, and G. Stamoulis. Traffic equivalence and substitution in a multiplexer with applications to dynamic available capacity estimation. *IEEE/ACM Transactions on Networking*, (2):217–231, 2002.

[30] S. Crosby. Statistical properties of a near-optimal measurement-based cac algorithm. In *Proceedings of the IEEE ATM '97 Workshop, Lisboa, Portugal, 1997.*, 1997.

[31] R.L Cruz. A calculus for network delay, part i: Network elements in isolation. *IEEE Transactions in Information Theory*, 37:114–131, 1991.

[32] R.L Cruz. A calculus for network delay, part ii: Network analysis. *IEEE Transactions in Information Theory*, 37:132–141, 1991.

[33] A. Dembo and O. Zeitouni. *Large Deviation Techniques and Applications.* Springer-Verlag, 1997.

[34] A. Demers, S.Keshlav, and S.Shenker. Analysis and simulation of a fair queuing algorithm. *Journal of Interworking Research and Experience*, pages 3–26, 1990.

[35] N. Duffield and all. Entropy of atm traffic streams: a tool for estimating qos parameters. *IEEE Journal of Selected Areas in Communications*, 13(6):981–990, 1995.

[36] N. G. Duffield. Asymptotic sampling properties of effective bandwidth estimation for admission control. In *Infocom*, 1999.

[37] N. G. Duffield, Pawan Goyal, Albert Greenberg, Partho Mishra, K. K. Ramakrishnan, and Jacobus E. van der Merwe. A flexible model for resource management in virtual private networks. In *ACM SIGCOMM'99*, 1999.

[38] Z. Dziong, M. Jurda, and L.G. Mason. A framework for bandwidth management in atm networks-aggregate equivalent bandwidth estimation approach. *IEEE Transactions in networking*, 5(1), 1997.

[39] B. Ionescu et al. A testbed and research network for next generation services over next generation networks. In *Tridentcom 2005*, 2005.

[40] Dolores M. Etter. *Introduction to Matlab 7.* Pearson eduction, 2005.

[41] W. Feng. A self configuring red gateway. In *Infocom*, 1999.

[42] W.-C. Feng, D. Kandlur, D. Saha, and K. Shin. Blue: A new class of active queue management algorithms. Technical Report CSE-TR-387-99, University of Michigan, April 1999.

[43] V. Firoiu and M. Borden. A study of active queue management for congestion control. In *Proceedings of IEEE INFOCOM'00*, volume 3, pages 1435–1444, Tel-Aviv, Israel, April 2000.

[44] S. Floyd. Tcp and explicit congestion notification. *ACM Computer Communication Review*, 24, 1994.

[45] S. Floyd and V. Jacobson. Random early detection for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1:397–413, 1993.

[46] TeleManagement Forum. SLA Management Handbook. Technical report, www.tmforum.org, 2001.

[47] G.F. Franklin, J.D. Powell, and M.L. Workman. *Digital Control of Dynamic Systems.* Addison-Wesley, 1990.

[48] M. Ge, M. Chiu, and Q. Wang. Robust pid controller design via lmi approach. *Journal of Process Control*, pages 3–13, 2002.

[49] A. Gersht and K.J. Lee. A congestion control framework for atm networks. *IEEE Journal on Selected Areas in Communications*, 9:1119–1130, 1991.

[50] R.J. Gibbens and F.P. Kelly. A decision theoretic approach to call admission control in atm networks. *IEEE JSAC*, 13(1), 1995.

[51] Natalie Giroux and Sudhaker Ganti. *Quality of Service in ATM networks.* Prentice Hall, December 1998.

[52] B. Gleeson and all. *A Framework for IP Based Virtual Private Networks.* Internet RFC 2764, 2000.

[53] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice Using MATLAB, 2nd Edition.* Wiley, January 2001.

[54] M. Grossglauser and N.C. Tse. A framework for robust measurement based admission control. *IEEE Transactions in Networking*, 7(3), 1999.

[55] R. Guerin and all. Equivalent capacity and its application to bandwidth allocation in high speed networks. *IEEE JSAC*, 9:968–981, 1991.

[56] M. Handley, S. Floyd, J. Pahdye, and J. Widmeer. Tcp friendly rate control (tfrc): Protocol specification. RFC 3448, January 2003.

[57] Simon Haykin and Barry Veen. *Signals and Systems.* Wiley, 2003.

[58] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. RFC 2597: Assured Forwarding PHB Group. Technical report, www.ietf.org, 1999.

[59] D. Heyman and T. Lakshman. What are the implications of long-range dependence for vbr traffic engineering. *IEEE/ACM Transactions in Networking*, 4:301–317, 1996.

[60] C. Hollot, V. Misra, D. Towsley, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *INFOCOM 2001*, 2001.

[61] C. V. Hollot, Vishal Misra, Donald F. Towsley, and Weibo Gong. A control theoretic analysis of RED. In *INFOCOM*, pages 1510–1519, 2001.

[62] ITU-T. Y.1311 - Network-based IP VPN over MPLS architecture. Technical report, www.itu.ch, 2001.

[63] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM*, August 1988.

[64] S. Jamin, P. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. In *SIGCOMM'95*, pages 2–13, 1995.

[65] S. Kalyanaraman and R. Jain. The ERICA switch algorithm for ABR traffic management in ATM networks. *IEEE/ACM Transactions in Networking*, 8:87–98, 2000.

[66] F. Kelly. Effective bandwidths at multi-class queues. *Queuing Systems*, 9:5–16, 1991.

[67] F. Kelly. Notes on effective bandwidths. In *Stochastic Networks: Theory and Applications*, volume Stochastic Networks: Theory and Applications, pages 141–168. Oxford University Press, 1996.

[68] Y. Kim and San qi Li. Timescale of interest in traffic measurement for link bandwidth allocation. In *Infocom*, pages 738–748, 1996.

[69] A. Kolarov and G. Ramamurthy. A Control Theoretic Approach to the Design of an Explicit Rate Controller for ABR Service. *IEEE/ACM Transactions on Networking*, 7:741–753, 1999.

[70] K. Kumar, G. Margrave, and D. Mitra. Novel Techniques for Design and Control of Generalized Processor Sharing Schedulers for Multiple QoS Classes. In *INFOCOM 2000*, 2000.

[71] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management. In *Proceedings of ACM SIGCOMM'01*, pages 123–134, San Diego, CA, August 2001.

[72] C. Lambiri. *On the Estimation and Control of Packet Loss For VPN Services.* PhD thesis, University of Ottawa, 2003.

[73] J.L. LeBoudec. *Network Calculus.* Springer-Verlag-LNCS 2050, 2001.

[74] N. Likhanov and R. R. Mazumdar. Cell loss asymptotics in buffer fed with a large number of independent stationary sources. In *INFOCOM 1998*, 1998.

[75] Lennart Ljung. *System Identification - Theory For the User.* Prentice Hall, 1999.

[76] Lennart Ljung and Torsten Soderstrom. *Theory and Practice of Recursive Identification.* The MIT Press, 1983.

[77] Lonnie C. Ludeman. *Random Process Filtering, Estimation, and Detecion.* John wiley, 2003.

[78] ITU-T Recommendations M.3010. *Principles for a Telecommunications Management Network.* ITU-T, 1996.

[79] K. McCloghrie and M. Rose. *Management Information Base for Network Management of TCP/IP-based internets: MIB-II.* IETF RFC 1213, 1991.

[80] M. Montgomery and G. de Veciana. On the relevance of time scales in performance oriented traffic characterizations. In *Infocom*, 1996.

[81] J. Nagle. On Packet Switches with Infinite Storage. *IEEE Transactions on Communications*, 35:435?38, 1987.

[82] Yu. Nesterov and A. Nemirovski. *Interior Point Polynomial methods in Convex Programming: Theory and Applications.* SIAM books, Philadelphia, 1994.

[83] Object Management Group. *The Common Object Request Broker: Architecture and Specification*, version 2.0 edition, July 1995.

[84] Teunis J. Ott, T. V. Lakshman, and Larry H. Wong. SRED: Stabilized RED. In *Proceedings of INFOCOM*, volume 3, pages 1346–1355, 1999.

[85] Randy Otte, Paul Patrick, and Mark Roy. *Understanding CORBA: The Common Object Request Broker Architecture.* Prentice Hall, 1996.

[86] S. Panchen P. Phaal and N. McKee. *InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks.* IETF RFC 3176, 2001.

[87] D. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot. Analysis of measured single-hop delay from an operational backbone network. June 2002.

[88] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Transactions on Networking*, 1:137–150, 1993.

[89] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Transactions on Networking*, 2:347–357, 1993.

[90] V. Paxon and S. Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Trans. Networking*, 3(3):226–244, June 1995.

[91] V. Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, 1999.

[92] Charles L. Phillips and H. Troy Nagle. *Digital Control System Analysis and Design.* Prentice Hall, 1995.

[93] J.B. Postel. *Transmission Control Protocol.* Internet RFC 793, September 1981.

[94] J. Qiu and E.W. Knightly. Measurement-based admission control with aggregate traffic envelopes. *IEEE Transactions in Networking*, 9(2), 2001.

[95] Martin Reisslein, Keith W. Ross, and Srinivas Rajagopal. A framework for guaranteeing statistical qos. *IEEE/ACM Transactions on Networking*, 10(1):27–42, 2002.

[96] J. Roberts and all. (eds). *Broadband Network Teletraffic - Performance Evaluation and Design of Broadband Multiservice Networks.* Springer-Verlag LNCS 1155, 1996.

[97] L. Roberts. Enhanced proportional rate control algorithm PRCA. ATM Forum Contribution 94-0735R1, August 1994.

[98] E. Rosen and Y. Rekhter. *BGP/MPLS VPNs.* www.ietf.org, 1999.

[99] D. Rosinova and V. Vesely. Robust static output feedback for discrete-time systems - lmi approach. *Periodica Polytechnica Ser. EL. ENG.*, pages 151–163, 2004.

[100] B. Ryu and A. Elwalid. The importance of long-range dependence of vbr video traffic in atm traffic engineering: Myths and realities. *Computer Communications Rev.*, 26:3–14, 1996.

[101] Y. Serberst and Sang gi Li. Unified measurement functions for traffic aggregation and link capacity assessment. In *INFOCOM*, pages 1522–1531, 1999.

[102] S. Shenker, C. Partridge, and R. Guérin. *Specification of Guaranteed Quality of Service.* IETF, September 1997.

[103] W. Stallings. *Data and Computer Communications.* Prentice Hall, 2003.

[104] T.E. Stern and A.I. Elwalid. Analysis of a separable markov modulated rate models for information handling systems. *Adv. Applied Prob.*, 23:105–139, 1992.

[105] W. Stevens. *TCP Slow start, congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms.* Internet RFC 2001, January 1997.

[106] Andrew Tanenbaum. *Computer Networks.* Prentice-Hall, 2003.

[107] D. Tse and M. Grosslauser. Measurement-based call admission control: Analysis and simulation. In *INFOCOM'97*, pages 981–989, 1997.

[108] D.N.C. Tse, R.G. Gallager, and J.N. Tsitsiklis. Statistical multiplexing of multiple time scale markov streams. *IEEE/ACM Transactions in Networking*, 1(6), 1995.

[109] B. Tsybakov and N. Georganas. On Self Similar Traffic in ATM Queues: Definitions, Overflow probability bound and cell delay distribution. *IEEE/ACM Transactions in Networking*, 5:397–409, 1997.

[110] B. Tsybakov and N. Georganas. Overflow Probability in an ATM Queue with Self-Similar Traffic . In *ICC'97 Montreal*, 1997.

[111] Ronald E. Walpole and Raymond H. Myers. *Probability and Statistics for Engineers and Scientists.* Prentice Hall, 1993.

[112] A. Weiss. A new technique of analyzing large traffic systems. *Advances in Applied Probability*, 18:506–532, 1986.

[113] J. Wroclawski. *Specification of the Controlled-Load Network Element Service.* IETF, September 1997.

[114] Dongli Zhang. A distributed ip flow monitoring scheme in heterogeneous high speed network environment. In *ICUE 2004*, Toronto, Canada.

[115] Dongli Zhang and Dan Ionescu. Measurement and control of packet loss probability for mpls vpn services. *IEEE Transaction on Instrumentation and Measurement*, 55(5):1587–1598, October 2006.

[116] Dongli Zhang and Dan Ionescu. Using reactive packet loss probability estimator for robust packet loss control for mpls vpn services. *Periodica Politechnica, Transactions on Automatic Control an Computer Science*, 2:7–12, June 2006.

[117] Dongli Zhang and Dan Ionescu. A new practical packet loss estimator for mpls vpn services. In *APCC 2004*, Beijing, China.

[118] Dongli Zhang and Dan Ionescu. On packet loss estimation for virtual private networks services. In *ICCCN 2004*, Chicago, IL, USA.

[119] Dongli Zhang and Dan Ionescu. A scalable qos control framework for next generation ip/mpls core network. In *ICCT 2006*, Guilin, China.

[120] Dongli Zhang and Dan Ionescu. Distributed measurement of packet loss probability with application to admission control of vpn services. In *IMTC 2005*, Ottawa, Canada.

[121] Dongli Zhang and Dan Ionescu. Packet loss measurement and control for vpn based services. In *IMTC 2005*, Ottawa, Canada.

[122] Dongli Zhang and Dan Ionescu. Robust packet loss control with an lmi approach. In *CCECE 2006*, Ottawa, Canada.

[123] Dongli Zhang and Dan Ionescu. Robust and optimal control of packet loss probability. In *Globecom 2006*, San Francisco, USA.

[124] Dongli Zhang and Dan Ionescu. Feedback control of packet loss for vpn services. In *ICON 2004*, Singapore.

[125] Dongli Zhang and Dan Ionescu. Packet loss probability measurement with a kalman filter approach. In *IMTC 2006*, Sorrento, Italy.

[126] Dongli Zhang and Dan Ionescu. Recursive parameter estimation for the dynamic packet loss system. In *Globecom 2005*, St. Louis, USA.

[127] Dongli Zhang and Dan Ionescu. Packet loss measurement and estimation for vpn based services. *IEEE Transaction on Instrumentation and Measurement*, submitted, 2005.

[128] Dongli Zhang and Dan Ionescu. On the estimation, identification and control of packet loss probability. *IEEE/ACM Transaction on Networking*, submitted, 2006.

[129] Dongli Zhang and Dan Ionescu. Reactive estimation of packet loss probability for vpn based services. *IEEE/ACM Transaction on Networking*, submitted, 2006.

[130] Dongli Zhang, Dan Ionescu, and Stejarel Veres. Identification of the dynamic packet loss system for the vpn based services. *IEEE Transaction on Communication*, submitted, 2006.

[131] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks . *IEEE/ACM Transactions in Networking*, 5:1374–1396, 1995.

[132] H. Zhang and D. Ferrari. Rate-controlled service disciplines. *Journal of High Speed Networks*, 3(4):389–412, 1994.

[133] L. Zhang, S. Deering, and D. Estrin. RSVP: A new resource ReSerVation protocol. *IEEE network*, 7(5), September 1993.

[134] K. Zhou and J.C. Doyle. *Essentials of Robust Control.* Prentice Hall, 1998.