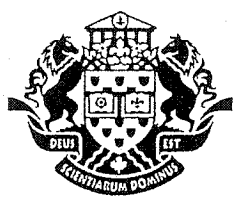

Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Cheng LI

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M. Sc. (System Science)

GRADE - DEGREE

System Science Program

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Fluid Model for Access Control Mechanism

N.U. Ahamed

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

O. Yang

T. Yeap

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Fluid Model for Access Control Mechanism

Cheng Li

A thesis submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the degree of
Master of Science

Systems Science,
University of Ottawa,
Ottawa, Ontario, Canada K1N 6N5

February 2004

Your file Votre référence
ISBN: 0-494-01526-8
Our file Notre référence
ISBN: 0-494-01526-8

Acknowledgements

I wish to express my sincere thanks to Dr. Nasir Uddin Ahmed, my advisor, for his continuous guidance and support. Without his patience and cooperation throughout this research, it would not have been possible to finish this thesis. My special thanks go to Dr. Xinhuo Hua, Ms. Hong Yan, Bo Li and Hui Song for their valuable comments and advice.

Dedication

To my wife Hong Zou, for her encouragements and support.

Abstract

In this thesis, we develop two distinct traffic models: one based on Brownian motion and the other based on fractional Brownian motion. The later model captures the self-similarity and the long-range dependence (LRD) properties. The aggregate model is composed of a drift part and of a fluctuation (diffusion) part. With this model, traffic from several seconds to 24 hours can be simulated.

Applying the Token Bucket (TB) mechanism, a continuous time (state-space) dynamic system model is developed based on the ideas of recent papers [1,2]. Incoming traffic from each user is policed at the TBs and one multiplexor buffer, linked to all the TBs, multiplexes the conforming traffic. We propose two feedback control strategies, one is a simple feedback control law and the other is a feedback control based on neural network, to control the traffic flow into the link of the backbone network. We also use the simulated annealing algorithm to optimize the parameters of control laws. Several network performance related issues are studied systemically. The results show that the proposed control laws can improve the network performance, by improving throughput, reducing multiplexor and TB losses, and relaxing, not avoiding, congestion.

Contents

List of Figures

List of Tables

Acronyms

CBR	Constant Bit Rate
DES	Discrete Event System
Diffserv	Differentiated Service
HJB	Hamilton-Jacobi-Bellman
EM	Euler-Maruyama
FB	Feed Back Control
FBM	Fraction Brownian Motion
FBC	Feedback Control Law (simple)
FBCN	Feedback Control Based on Neural Network
GA	Generic Algorithm
IETF	Internet Engineering Task Force
Intserv	Integrated Service
ISP	Internet Service Provider
LRD	Long Range Dependence
MC	Monte Carlo Method
MPEG	Moving Picture Experts Group
NN	Neural Network
OPM	Open Loop Control with Mean Traffic Rate
OPP	Open Loop Control with Peak Traffic Rate
QoS	Quality of Service

RSVP	Resource Reservation Protocol
SA	Simulated Annealing
SDE	Stochastic Differential Equation
SRD	Short Range Dependence
TB	Token Bucket
VBR	Variable Bit Rate
VTP	Variance Time Plot

Symbols

A	Variation of $c(t)$
$a_i(t)$	i th link incoming traffic rate at time t
$\rho_i(t)$	i th token bucket state at t
c	Network link capacity (Constant)
$c(t)$	Network link capacity (variable)
f	Frequency of $c(t)$
J	Objective function
K	Number of time intervals in a trace
M	Number of sample paths used
$q(t)$	Multiplexor buffer state at time t
r	A constant that determines the variance of the generating traffic
Q	Multiplexor buffer capacity
T_i	i th token bucket size
$u_i(t)$	i th tokens generating rate at time t
$\lambda_1(t)$	Weight given to multiplexor losses
$\lambda_2(t)$	Weight given to TB losses
$\lambda_3(t)$	Weight given to waiting loss
Δt	Time interval

Chapter 1

Introduction

1.1 Motivation

Due to the approximately infinite user population, the large correlation of user demand, and the peak load that can be orders of magnitude greater than average, overload becomes an inevitable aspect for systems connected to the Internet. It is well known that the Internet does not provide Quality of Service (QoS). The widely used IP protocol offers best effort service, it does not guarantee bandwidth, delay and packet loss. As IP networks have been extensively deployed, all QoS efforts have to be layered on top of the existing technology. Network architectures such as Integrated Service (Intserv) and Differentiated Service (Diffserv), which are proposed by the Internet Engineering Task Force (IETF), provide QoS that delivers real time applications effectively and efficiently. The proliferation of the Internet and its excessive congestion has led researchers working on emerging high-speed telecommunication networks to develop tools to police and control the traffic at the user or source end [8]. Token bucket algorithm (TB), which is used as a filter to characterize traffic, has been employed in traffic control including admission control, access control, and flow control [3-6].

A TB has two parameters: its size T and token generating rate u. TB is operated according to the following rules: Tokens are placed in the bucket by the operating system of the router at a certain rate, new tokens are discarded if the bucket is full. An arriving packet of size S consumes S tokens. If there are not enough tokens, the packet is either discarded or marked. In the long run, the conformed traffic rate is limited to u though a burst of size T can be admitted.

Applied as a policing mechanism, a TB is essentially a credit management mechanism that controls the traffic entering the network [9][10]. This mechanism is widely applied on edge routers, such as Cisco 2600, 3600, 4500, 4700, and 7200 series routers, to police incoming or outgoing traffic [11]. Cisco edge routers use constant token generating rates but modify the TB algorithm by allowing compounded debt, which is a certain number of tokens that the system could borrow when token bucket contains fewer tokens than required. With this improvement, the system extends its burst capability. The system allocates each access link a given proportion of its total traffic transfer capacity. However, since incoming traffic rates may change arbitrarily, by using a constant token generating rate, the system cannot react to this change efficiently and dynamically.

To manage traffic in Internet services, a feedback-driven control is a right approach, by which the system actively observes its behavior and performance, and employs dynamic control to allocate resources [12]. A new dynamic model, in which all TBs linked to one multiplexor (queue), was proposed in [2]. Traffic policing, multiplexing and network utilization were formally defined in the model. To improve QoS and network utilization, some feedback control laws were presented. The research in [2] provided a systematic tool and guideline for the study of TB. However, there are some limitations that have to be improved and completed. In particular, the following aspects have to be addressed:

- The feedback control laws were not discussed in enough depth.

- Although the concept of a traffic model is proposed in [2], it is just demonstrated with a given deterministic traffic, the MPEG video trace, as its input traffic.

In [13], dynamic programming and Generic Algorithm (GA) were applied to optimize the control of deterministic MPEG video traces and obtained some good results. However, for stochastic incoming traffic, this method may not be practical.

The model in [2] was further improved by [14], where the details of the feedback control law were considered, and different kinds of stochastic traffic were used in the system to evaluate the system performance. The numerical results demonstrated that this system could be adapted to any kind of stochastic traffic. However, parameters of the feedback control law in [14] were not optimized. Moreover, all models in [2][13][14] were discrete time models, which studied lower traffic rates and were on a packet level.

Nowadays data is sent at a higher rate. So if we consider too much detail on the packet level, the system may be very complicated. To study the overall control performance of the system, some researchers [1][8][15][16][17] built fluid models that were continuous in time or in states. It is certainly a good approximation to discrete event systems (DES) for very high speed data flows. A TB fluid feedback control model with buffers was proposed in [17]. The result shows that a fluid flow model can accurately describe the behavior of TB systems, however, the systems were not analyzed theoretically. Authors of the paper [1] developed continuous time dynamic models based on the token bucket control mechanism, and they proved the existence and regularity properties of solutions for those systems. Although mathematical models were proposed in [1], the authors were not sure whether those models were appropriate for a stochastic process.

We think it would be very interesting to realize the continuous time mathematical models, propose efficient control strategies for dynamic stochastic traffic, and evaluate system

performances.

Making use of a real network to test models and algorithms is expensive, and sometimes impossible. In general, traditional discrete event simulations work well. However, even the most efficiently coded simulators suffer from the problem of scaling [16]. Since we will model the system based on the fluid concept, it is necessary to have a continuous dynamic traffic generator that can describe traffic characteristics. Due to the size, complexity, diversity, and rapid change of networks, the traffic flow is complicated, and its simulation is very difficult.

The Poisson process model has been proved to be accurate for modelling human generated actions, such as ftp session arrivals. But the Poisson process model cannot describe some data traffic characteristics. As we know, burstiness is observed at many time scales, the Poisson process model does not illustrate this characteristic. Burstiness of data traffic increases with the increment of traffic volume. The Poisson process model tends to smooth the burstiness as the traffic is heavier [18].

Recent network traffic measurement studies have shown the existence of self similarity and long range dependency (LRD) properties [19-24] (see Section 2.6 for details). The self similarity means that the same statistical patterns of behavior occur over time scales which can vary by many orders of magnitude. The LRD exhibits slowly decaying autocorrelation.

Standard Brownian motion and fractional Brownian motion (Section 2.7), which are stochastic processes, have self-similar characteristic. When the so called Hurst parameter H lies in the range $0.5 < H < 1$, Fractional Brownian motion also has the LRD property. It is natural to make use of these properties in network traffic simulation.

In order to obtain realistic traffic models for self-similar traffic, recent researches have been conducted. Most of them are stochastic models based on Brownian motion [25]. Time domain approach and frequency domain approach are two existing techniques for the generation of self-similar traffic [26]. Both of them have a drawback: they may produce negative values. To avoid negative value traffic, some techniques are applied. For instance, Kushner [27] added an extra item in the traffic generating function, this item increases when other items in the function try to drive the function negative. However, this method is not sufficiently reasonable, since it creates another problem: how the extra item increases. Therefore, a better simulation method associated with the mathematical models is required.

1.2 Objective

Based on our motivation above, we are focusing on the following points in this thesis:

1. to simulate traffic that can capture major properties of network traffic: LRD and self similarity;

2. to implement the mathematical model given in the paper [1] and propose new access feedback control laws which can improve the system performance and are easy to implement;

3. to use proper methods to optimize the parameters in the control laws in order to obtain the near minimal cost functional.

1.3 Thesis Organization

In this thesis, we will start with reviewing basic concepts related to this thesis (Chapter 2). Traffic models and a system model are given in Chapter 3. Objective function and other performance measures, which are used to evaluate the system, are defined in Chapter 4. In Chapter 5, we will discuss control strategies, propose two feedback control laws: a simple feedback control, and a feedback control based on neural network. Chapter 6 introduces numerical methods used in this thesis. Basic data used and assumptions for simulation experiments are presented in Chapter 7. We also analyze numerical results in Chapter 8. The thesis ends with a conclusion and some directions for further research (Chapter 9).

1.4 Contributions

In this thesis we describe and implement traffic simulation models and system model. We also propose and evaluate two feedback control strategies. This thesis makes four major contributions:

- This thesis makes use of fractional Brownian motion and standard Brownian motion to build traffic generating models, which capture real traffic LRD and self similarity characteristics.
- We realize the mathematical model proposed in the paper [1];
- We propose two control strategies: one feedback control law considering the multiplexor state and the change rate of the state, another feedback control based on neural network. The parameters of the feedback control laws are optimized with simulated annealing algorithm. The numerical results demonstrate the great improvement of the system performance;

- An adaptive step method is implemented in a continuous model to avoid generating negative values.

Chapter 2

Basic Concepts

There are many concepts related to networks traffic. In this chapter, we introduce some basic concepts, which are the foundation of our traffic simulation and access control, involved in the thesis as background knowledge. Characteristics of traffic are introduced in Section 1. Section 2 gives the concept of quality of service. Since congestion is discussed in several chapters in this thesis, we review the basic idea about congestion control in Section 3. Token bucket algorithm applied in traffic regulation, traffic policing and shaping is briefly described in Section 4 and 5. The definition of self similarity is given in Section 6. We then introduce Brownian motion in Section 7. In Section 8, we describe Monte carlo method which is used in experiments. The Hurst parameter is to measure the degree of traffic self similarity. In Section 9, three Hurst parameter estimation methods are introduced and compared. Simulated annealing algorithm is a method to find global maxima or minimum value, it is applied to optimize the parameters of the feedback control law in the thesis later on, it is necessary to overview the principals (Section 10). In Section 11, Neural network is briefly described, this method is applied in the second feedback control law in Chapter 6.

2.1 Traffic Characteristics

The Internet is very complex. Due to the large size of the Internet, diverse network behaves, a variety of network topologies, the new link-level technologies, variations in routers, and the introduction of new protocols, traffic properties are difficult to characterize. Even though, after years study, people are still finding some characteristics [28]:

- 24-hour cycles in traffic patterns;
- Poisson arrivals for start times of user sessions;
- Self-similarity in traffic patterns;
- Heterogeneity and change.

24-hour cycles in traffic patterns

It has been found that network traffic has daily patterns. These patterns relate to human activity. Traffic volumes start to rise around 9 AM, reach a peak around 11 AM, decline after 5 PM as the business day ends. We will discuss the patterns in detail in chapter 3.

Poisson arrivals for start times of user sessions

Poisson processes has been studied for fifty years, and it has been proved accurate for human related activity such as telephone or ftp traffic. User sessions relate to activity when human decide to network. Poisson processes can also describe network user session arrivals very well. Different network arrival processes were tested in [30], and the result provided solid evidence to prove user sessions start times are Poisson processes.

Self-similarity in traffic patterns

Recent network traffic measurement studies have shown the existence of fractal or self similarity properties [19-24], which means that the same statistical patterns of behavior occur over time scales which can vary by many orders of magnitude. The effect of

self-similarity is to introduce long range correlation into the traffic stream, which is a phenomenon that is observed in practice. We will introduce self similarity in details in section 2.6.

Heterogeneity and change

Network topology, infrastructures, applications, routing activity, and protocols change dramatically over time, network traffic must keep changing and be heterogeneous.

2.2 Quality of Service

With the wider use of the Internet, more and more applications are conveyed through it, some are mission-critical applications (shown as Table 2.1), and they require the corresponding network to provide a variety of services. However, the Internet Protocol (IP) is a best-effort protocol, which cannot guarantee the data packets delivery. Quality of service (QoS) is proposed to offer differing levels of service to different requirements. QoS allows Internet service providers (ISP) to deploy bandwidth to intensity applications as video-conferencing and timing sensitive applications as real time transactions while making use of network links efficiently.

Availability, latency, jitter, and capacity are considered as four most of the important factors in terms of impacting quality of a network connection. Availability is the percentage of a network's available time. Latency is the time taken by data to travel from the source to the destination. Jitter is the variance of latencies. Capacity is the total available bandwidth on a link.

The integrated services model (IntServ) and the differentiated services model (DiffServ)

are two main approaches to QoS. IntServ negotiates a particular QoS on a per flow basis. Before transferring traffic, a particular QoS level should be requested by the sender and receiver from the network. Once accepted, the intermediate routers keep the traffic flow at a specific level of jitter, latency, and capacity. Resource Reservation Protocol (RSVP), which is used for signaling QoS requirements for a particular traffic flow, is a key component in this model. DiffServ is a different approach. The goal of DiffServ is to offer differing levels of QoS to different users. The traffic entering the network is classified into a small number of classes. Each class is characterized with a tag, a router services packets according to the tags. When the sender needs a particular kind of service, it marks the individual packets accordingly.

Traffic Type	**Bandwidth Sensitivities**	**Loss Sensitivities**	**Delay Sensitivities**	**Jitter Sensitivities**
Voice	Very low	Medium	High	High
E-commerce	Low	High	High	Low
Transactions	Low	High	High	Low
E-mail	Low	High	Low	Low
Telnet	Low	High	Medium	Low
Casual browsing	Low	Medium	Medium	Low
Serious browsing	Medium	High	High	Low
File transfers	High	Medium	Low	Low
Video Conferencing	High	Medium	High	High
Multicasting	High	High	High	High

Table 2.1: Applications and Their QoS Requirements

2.3 Congestion Control

The network is in the state of congestion if the aggregate demand for bandwidth exceeds the available capacity. Congestion is one of most significant problems in Internet traffic engineering. When it occurs, networks' performance degrades.

Congestion can be classified based on time scale:

- Long period congestion (weeks to months);
- Medium period congestion (minutes to days);
- Short period congestion (Pico seconds to minutes).

The objective of congestion control is to keep the volume of traffic in the network below the level at which performance drops dramatically. Generally, there are two solutions to solve congestion problem: increasing the network capacity or reducing the traffic load.

Different congestion control policies deal with different types of congestions. In order to control a short period congestion, router mechanisms are used to adaptively regulate the traffic rate sent into the network. Re-configuring logical topology or adjusting some router parameters are medium period congestion control policies. Hardware update is usually carried out to solve a long period congestion.

Congestion control can also be classified based on reactive versus preventive: congestion recovery, congestion avoidance. Congestion recovery reacts to existing congestion, while congestion avoidance works on preventing congestion occurrence [31].

2.4 Token Bucket Algorithm in Traffic Regulation

A token bucket, which is formal definition of a transfer rate, is used to regulate the data in a flow. A token bucket is characterized by two parameters: the token generation rate at which tokens are created and placed in the bucket, and the bucket depth which is the token holding capacity. Tokens are generated and put into bucket at a certain rate. A bucket has a specified capacity. If tokens accumulated in a bucket reaches its bucket capacity, newly arriving tokens are rejected. Each token is a permission for the source to send a certain number of bits into the network. To send a packet, the regulator must remove a number of tokens equal in representation to the packet size from the bucket. If there are not enough tokens for arrival traffic, those non-conforming packets will be dropped, marked or buffered depending on how the token bucket is being used.

According to the definition, in any integral multiple-interval, the interface bit rate of the network cannot exceed the mean rate. However, the bit rate may be arbitrarily larger within the interval. If a token bucket can hold b tokens; token are generated at a rate of r token/sec unless the bucket is full of tokens. Over an interval of length t, the number of packets that are admitted is less than or equal to $(rt + b)$. The token bucket mechanism is implemented in traffic policing and traffic shaping.

2.5 Traffic Policing and Shaping

Traffic policing (Figure 2.1) and shaping (Figure 2.2) utilize the token bucket mechanism to regulate traffic. A token bucket system that has both a token bucket and a data buffer is applied for traffic shaping. If it only has a token bucket, it would be used for traffic policing. For traffic shaping, the arrival packets that cannot be conformed immediately are delayed in the data buffer.

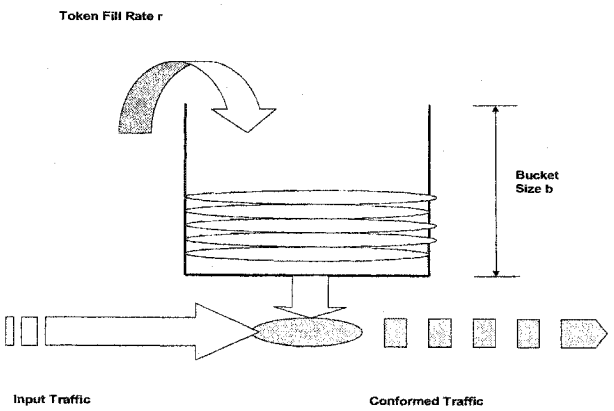

Figure 2.1: Token Bucket Type I

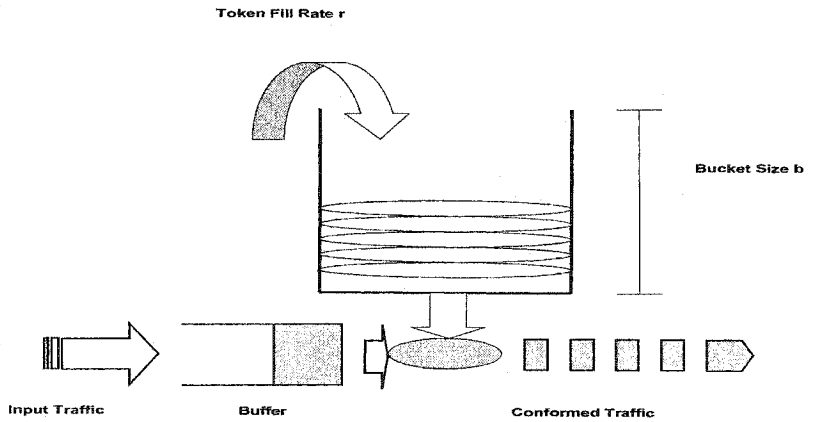

Figure 2.2: Token Bucket Type II

Network administrators use traffic policing to limit application traffic to the configured rates on the router (shown as Figure 2.3 [32]). Traffic policing is often configured on interfaces at the edge of a network to limit traffic into or out of the network. By buffering the bursts at the network edges, traffic shaping can smooth out application bursts (shown as Figure 2.4 [32]), so that network congestion is reduced to acceptable levels.

Traffic policing and shaping usually use the same way to identify traffic descriptor violations. The difference between them is the way they respond to violations. For instance:

- Policing typically drops traffic.
- Shaping typically uses a buffer, or queuing mechanism to delay excess traffic, hold packets and shape the flow when the source data rate exceeds expected;

Traffic shaping and policing can be used separately or in tandem. We place emphasis on traffic policing in this thesis.

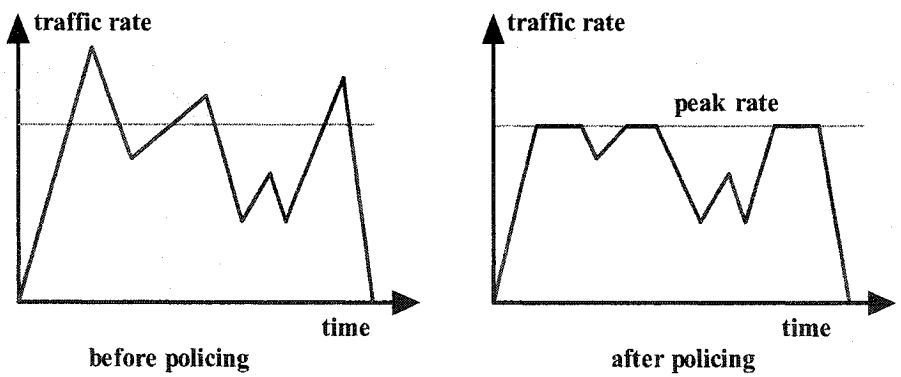

Figure 2.3: Policing Function

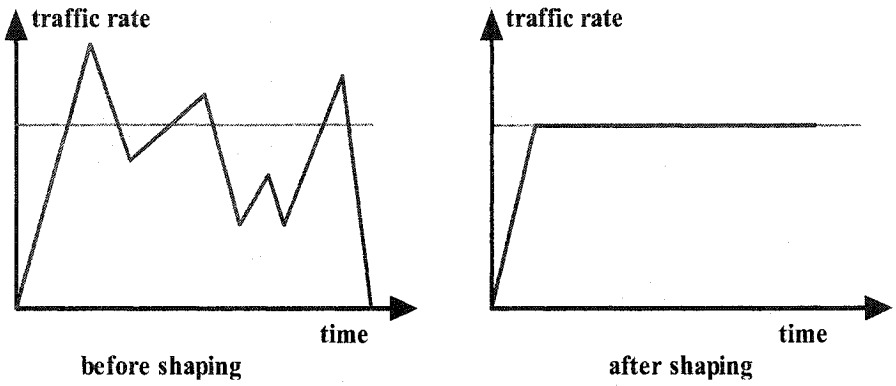

Figure 2.4: Shaping Function

2.6 Self-Similarity

A stochastic process $X(t)_{t\in R_+}$ is said to be H-self similarity if $X(at) \stackrel{d}{=} a^H X(t)$ for any positive a. The equality $\stackrel{d}{=}$ means that two processes $X(at)$, $a^H X(t)$ have identical distribution.

Self-similar processes show long-range dependence. A process with long-range dependence has an autocorrelation function. The autocorrelation function of such a process decays hyperbolically (as compared to the exponential decay exhibited by traditional traffic models). Hyperbolic decay is much slower than exponential decay, the sum of the autocorrelation values of such a series approaches infinity.

One of the attractive features of using self-similar models for time series, when appropriate, is that the degree of self-similarity of a series is expressed using only a single parameter. The parameter expresses the speed of decay of the series' autocorrelation function. For historical reasons, the parameter used is the Hurst parameter H. Thus, for self-similar processes, $0.5 < H < 1$, as $H \rightarrow 1$, the degree of self-similarity increases.

2.7 Brownian Motion

Pollen grains suspended in water move in a zigzag path under the microscope. This phenomenon was first observed by Robert Brown in 1827. Indeed, liquid molecules at all directions keep bombarding a suspended particle randomly. If the particle is very small, the hits that it takes from one side are stronger than the bumps from other sides, causing it to jump. These small random jumps are what make up Brownian motion [49]. There are two kinds of Brownian motion: standard Brownian motion and fractional Brownian motion.

Standard Brownian Motion

A real valued process is called a standard Brownian or Wiener motion, if the following conditions are satisfied:

1) $P\{W(0) = 0\} = 1$;

2) The increments of W given by $W(t_{i+1}) - W(t_i)$ on disjoint intervals of time $(t_i, t_{i+1}] \subset I$ are independent Gaussian random variables with mean zero and variance $(t_{i+1} - t_i)$ [33].

Fractional Brownian Motion

Fractional Brownian motion denoted by B^H, $t \geqslant 0$ was originally defined and studied by Kolmogorov. With a Hurst parameter $H \in (0; 1)$, it is a centered Gaussian process B^H, $t \geqslant 0$ with covariance

$$Cov[B_t^H \; B_s^H] = \frac{1}{2}(t^{2H} + s^{2H} - \mid t - s \mid^{2H}), \quad where \;\; (s, t \geq 0)$$

This process has stationary increments $Cov[(B^H(t) - B^H(s))^2] = |t - s|^{2H}$, where $(s, t) \geq 0)$, and is H-self similar. Indeed, for $H > 0.5$, the increments are positively correlated, and for $H < 0.5$, they are negatively correlated. It is useful to note that for $H > 0.5$, B^H is a long memory process: the covariance of increments at distance u decrease as u^{2H-2}.

Since network traffic is observed having long range dependence and self similarity characteristics, these significant properties of fractional Brownian motion make it a natural candidate as a model of noise in communication networks [34] (see, for instance, Leland, Taqqu and Willinger [20]).

For $H = 0.5$, fractional Brownian motion is reduced to standard Brownian motion. The main difference between them is that while the increments in the later process are independent, the increments are dependent in the former process. This dependence means that if there is an increasing pattern in the previous steps, then it is likely that the current step will increase as well.

2.8 Hurst Parameter Estimation

The Hurst parameter H measures traffic burstiness and long range dependence (LRD). It ranges between 0 and 1. The network traffic, with $1/2 < H < 1$, shows both self similarity and long range dependence. To simulate real traffic, we can estimate the parameter H, and then simulate traffic based on the estimated value.

The following three methods are usually used to estimate the parameter H:

- Periodogram plot;
- RS-statistic plot;
- Variance-time plot (VTP).

H	**Periodogram**	**RS-statistic**	**VTP**
0.5	40.7%	9.9%	12.9%
0.6	24.4%	9.7%	5.5%
0.7	15.6%	2.2%	3.3%
0.8	6.9%	1.5%	0.6%
0.9	1.5%	1.7%	1.0%

Table 2.2 Inaccuracy of Hurst Parameter Estimated Methods

For different hurst parameter values and methods, the corresponding inaccuracy of the hurst parameter is listed in Table 2.2 [29]. The Periodogram plot is the most inaccurate method. Comparing the VTP and the RS-statistic plot, the VTP gives more accurate results than that of the RS-statistic plot except $H = 0.5,\ 0.7$. Since Internet traffic has LRD property, for real traffic, the hurst parameter H is usually larger than 0.5, for $H \in (0.6, 0, 9)$, overall, the VTP is a better one. Moreover, the VTP is the fastest method in term of the computing time [30].

Based on the VTP, the parameter H is estimated in the following way [30]:
Let X_t represent the volume (bytes) of arrival packets at the $t - th$ time interval. For each $m = 1, 2, 3, ..., X^{(m)} = (X_k^{(m)} : k = 1, 2, 3, ...)$ denote the new covariance stationary time series obtained by averaging the original series X non-overlapping blocks of size m. For each $m = 1, 2, 3, ..., X_k^{(m)}$ is given by:

$$X_k^{(m)} = \frac{1}{m}(X_{km-m+1} + ... + X_{km}), \quad k \geq 1$$

According to the above formula, after all $X_k^{(m)}$ values are calculated, we can plot a graph:

$$f(m) = (log(m), log(Var(X^{(m)})))$$

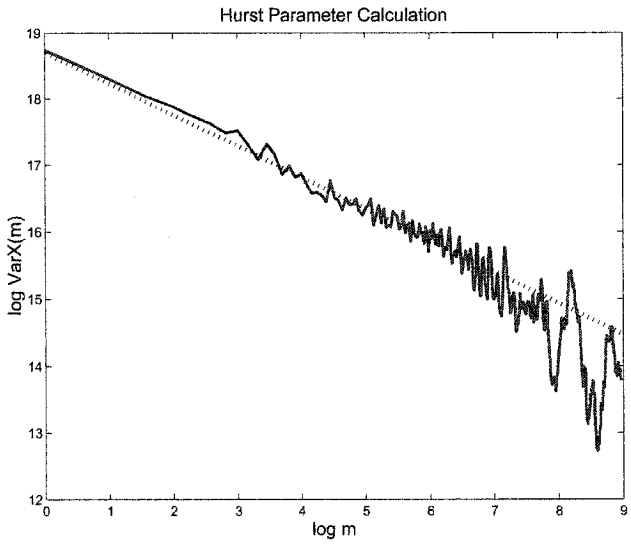

Figure 2.5: Hurst Parameter Estimation

Let the target slope of the graph is sl (the slope of the dotted line illustrated in Figure 2.5), and $\beta = -sl$, we can obtain the value of the hurst parameter $H = 1 - \beta/2$. The data of Figure 2.5 is 4.8 seconds Bellcore real traffic downloaded from [31], by applying the VTP, we get the parameter H of this trace of traffic is 0.7625.

2.9 Monte Carlo Method

Monte Carlo method (MC) is applied to calculate expected values throughout this thesis. We briefly introduce it here.

In 1944, a roulette, a simple random number generator, was created in Monte Carlo of Monaco. The method is called after name of the city. Its systematic development started since then [36]. With this technique, mathematical and physical problems, which cannot be evaluated analytically or is too complex to be calculated easily, can be approximately

solved by simulating random quantities. In this thesis, we will use the Monte Carlo integration, one of the most common applications of the Monte Carlo method, to evaluate expected values of random variables and functions of random processes.

Consider an arbitrary function $f(x)$, its expected value (mean) can be calculated with

$$< F >= \int_a^b f(x)p(x)dx,$$

where $p(x)$ is a p.d.f such that $p(x) \geq 0$ on $[a,\ b]$ and $\int_a^b p(x)dx = 1$. However, for some problems, it is difficult to achieve directly. So at first we compute the following sample mean, which is a Monte Carlo estimator of the expected value:

$$\overline{F}_N = \frac{1}{N}\sum_{i=1}^{N} f(x_i),$$

where $x_i \in [a, b]$. An alternate and more efficient approach is to select all the x_i randomly from a given probability distribution, a conventional option would be uniform grids.

By the Strong Law of Large Numbers, the limit $\overline{F}_N$ almost surely exists. Then according to the fundamental theorem of Monte Carlo, we get

$$< F >= \lim_{N\to\infty} \overline{F}_N$$

.

2.10 Simulated Annealing

Simulated annealing (SA) is a Monte Carlo approach for minimizing multivariate functions. Its major advantage over other methods is its ability to avoid becoming trapped at local minima. Since we apply this algorithm to optimize parameters of feedback control laws, it is necessary to present the basic conception.

SA was originally proposed by Metropolis in 1953 [35]. It is motivated by an analogy to annealing in solids. When a solid is heated past melting point and then is cooled, the structural properties of the solid depend on the rate of cooling. If the temperature of the liquid is lowered slowly enough, large crystals will be formed. The algorithm simulates the cooling process by gradually lowering the temperature of the system until it converges to a steady, frozen state. In 1982, Kirkpatrick et al [38] took the idea of the algorithm and applied it to search for feasible solutions and converge to an optimal solution.

Starting from an initial point, taking a step, the algorithm evaluates the cost function. While minimizing a function, it accepts any downhill step and the process iterates from this new point. It also may accept an uphill step with the probability based on the Metropolis criteria. As a result, it can jump out of local optima. With the optimization process proceeding, the algorithm closes to the global optimum. Since there are very few assumptions made regarding the cost function to be optimized, the algorithm is very robust with respect to non-quadratic surfaces [40].

To apply the simulated annealing, as it is illustrated in Figure 2.6 [39], a system is initialized with a particular configuration. A new configuration is constructed by imposing a random displacement. A sequence of iterations is generated. In each iteration, corresponding change ΔE in cost function is calculated. If the energy of this new state is lower than that of the previous one, the change is accepted unconditionally and the system is updated. If the energy is greater, the new configuration is accepted probabilistically.

The probability P_{acc} is based upon the Boltzmann distribution:

$$P_{acc}(\Delta E) = exp(-\frac{\Delta E}{kT})$$

where k is a constant and the temperature T is a control parameter.

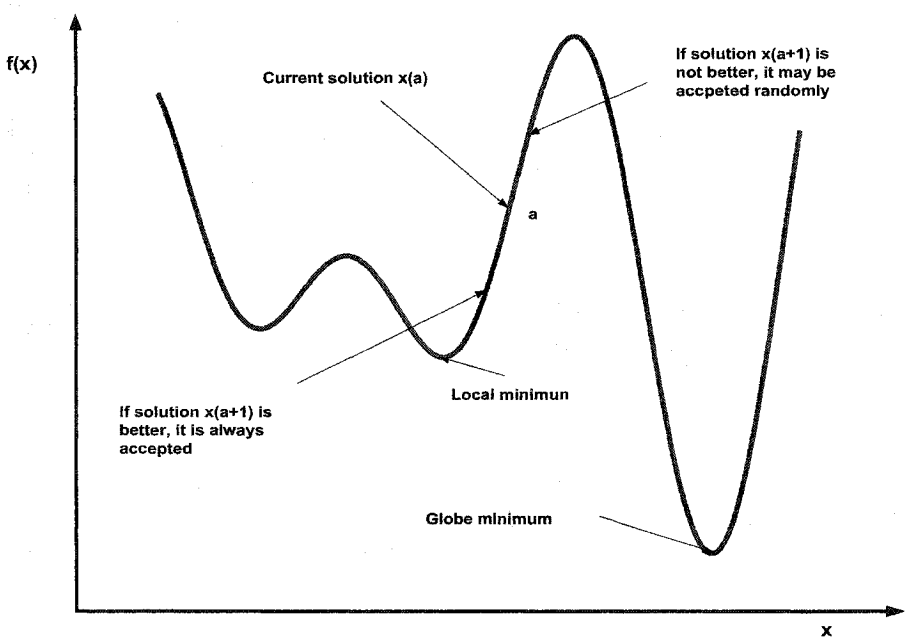

Figure 2.6: Simulated Annealing Principle

The condition of accepting a worse state is given by the equation:

$$P_{acc}(\Delta E) > r_u$$

where r_u is a uniformly distributed random number between 0 and 1.

This is the Metropolis step, the fundamental procedure of simulated annealing. This procedure allows the system to move consistently towards lower energy states, yet still jump out of local minima due to the probabilistic acceptance of some upward moves.

The way in which the temperature is decreased is known as the cooling schedule. it consists of four components: starting temperature, final temperature, temperature decrement, and a number of iterations at each temperature.

2.11 Neural Network

Neural network (NN) is employed to approximate feedback control in Chapter 5 of the thesis, we give its basic conception here.

NN is a system loosely based on the human brain. The field goes by many names, such as parallel distributed processing, neuro-computing, natural intelligent systems, machine learning algorithms, and artificial neural networks. It is an attempt to simulate within specialized hardware or sophisticated software, the multiple layers of simple processing elements called neurons. Each neuron is linked to a certain number of its neighbors with varying coefficients of connectivity that represent the strengths of these connections (a simple neural network is illustrated as Figure 2.7). Learning is accomplished by adjusting these strengths to cause the overall network to output appropriate results [41].

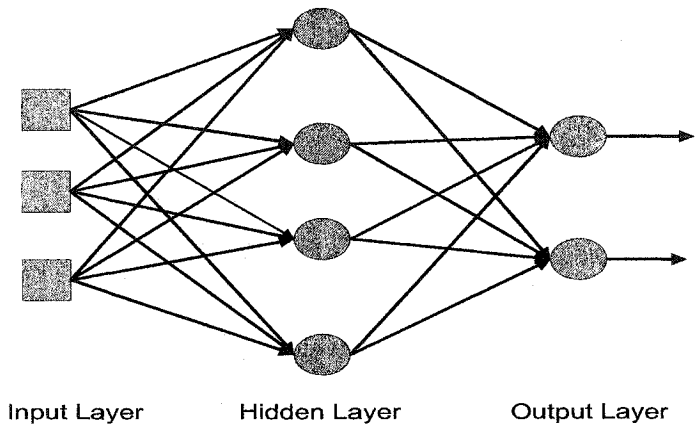

Figure 2.7: Feed-forward Neural Networks

NN was proposed in the late 1950's, and became sophisticated enough for general applications in the mid-1980's. Today an increasing number of real-world problems of considerable complexity are solved by applying it. It is powerful for pattern recognition, functional prediction and system modelling where the physical processes are not understood or are highly complex. The advantage of NN lies in its resilience against distortions in the input data and its capability of learning. it is often good at solving problems that are too complex for conventional technologies (e.g., problems that do not have an algorithmic solution or for which an algorithmic solution is too complex to be found).

Chapter 3

System Model

We consider a network scenario that is illustrated in Figure 3.1 as following [2]: a number of sources (each source consists many individual users aggregated) share one multiplexor in an access node that has a finite buffer and a certain service rate. Each of the users' traffic is processed by a dedicated token bucket before it is passed on to the multiplexor, where they are queued up for service into the out going link. There is a controller at the access node dynamically controlling the access rate of each traffic by adjusting the token supply. The system, which we consider in this thesis, includes the TBs and the multiplexor. In this chapter, we first introduce traffic models that are based on standard Brownian motion and fractional Brownian motion. Then we extend to a system model. Besides traffic models, TB population and multiplexor state are two important parts of the system, their equations are given in the second section.

The indicator function is used throughout the thesis, it is necessary to define the indicator function at the beginning. Let S denote any logical or mathematical statement, the indicator function is as follows:

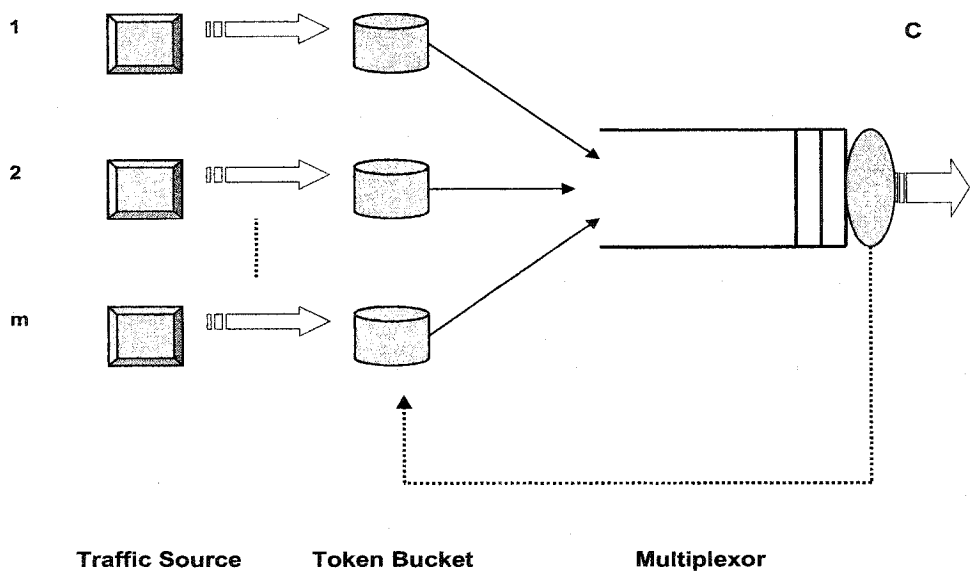

Figure 3.1: System Model

$$I(S) = \begin{cases} 1, & \text{if the statement S is true} \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

3.1 Traffic Models

We propose two traffic models based on standard Brownian motion and fractional Brownain motion respectively. In general, the traffic models are composed by a drift component and a diffusion component. In this section, we introduce the traffic models and discuss how to evaluate the related parameters.

3.1.1 Traffic Model Based on Standard Brownian Motion

We develop a traffic simulation model based on standard Brownian motion.

$$\begin{aligned} da_i = \quad & h_i(t)\{I(a_i(t) > 0, h_i(t) \leq 0) + I(h_i(t) > 0)\}dt \\ & +\sigma_i I\{a_i(t) > 0\}dW_i(t), \quad i = 1, 2, \cdots, m \end{aligned} \tag{3.2}$$

where $a_i(t)$ represents the i th incoming traffic rate, $h_i(t)$ is the i th traffic drift function, σ_i is a function of a diffusion component, $W_i(t)$ is standard Brownian motion.

This equation can be explained as follows. The change of the traffic rate is given by the sum of two items on the right hand side of the equation. The first item is a deterministic function $h_i(t)$ and its indicators. This item is the infinitesimal mean of the traffic rate, it determines general traffic change trends. It works when $h_i(t) > 0$ or $h_i(t) \leq 0$ but $a_i(t) > 0$. The second item is a infinitesimal variance-covariance matrix. It determines the volatility of the incoming traffic. This item affects the change of the traffic rate if $a_i(t) > 0$.

3.1.2 Traffic Model Based on Fractional Brownian Motion

To create a traffic model based on fractional Brownian motion, (3.2) is modified by the following equation:

$$\begin{aligned} da_i = \quad & h_i(t)\{I(a_i(t) > 0, h_i(t) \leq 0) + I(h_i(t) > 0)\}dt \\ & +\sigma_i I\{a_i(t) > 0\}dB_i^H(t), \quad i = 1, 2, \cdots, m \end{aligned} \tag{3.3}$$

In this equation, $W_i(t)$ of (3.2) is replaced by $B_i^H(t)$, which is fractional Brownian motion associated with the Hurst parameter (self-similarity parameter) H. The main difference between fractional Brownian motion and standard Brownian motion is that the later has independent increments, while the former does not have this property.

As we mentioned in Section 2.6, the Hurst parameter is the one to describe the degree of LRD and the burstiness of the traffic. The range of the parameter H is between 0 and 1. Fractional Brownian motion turns into standard Brownian motion for $H = 0.5$. If $H \in (0.5, 1)$, B^H has self similarity and long range dependence properties, which are characteristics of Internet traffic.

Equations (3.2) or (3.3) give the source dynamics representing the traffic rate generated by the i-th user. In case of multiple users, say, m, the scalar a_i is replaced by an m vector $a \equiv \{a_i,\ i = 1, 2 \cdots m\}$.

Thus the source dynamics is governed by the following system of stochastic differential equation of Ito type:

$$da = b(t, a)dt + \sigma(t, a)dB \tag{3.4}$$

where B represents either the standard Brownian motion or the FBM. Since, in general, the sources are independent users, the drift vector b is the vector of infinitesimal rates of

individual users and the diffusion (fluctuation/voltility) σ is a diagonal matrix representing the intensity of fluctuation of each of the users individual traffic.

3.1.3 Basic Parameters in Traffic Models

The function $h(t)$ of the drift component, the function $\sigma(t)$ and the Hurst parameter H of the diffusion component are involved in our traffic models. Here we discuss how to determine them.

Evaluation of Drift Parameter

As a deterministic function, $h(t)$ restricts the traffic trend. For a short period of time, such as a few seconds, traffic volume may change arbitrarily. However, $h(t)$ exists and can be observed in a long period of time. For instance, traffic volume follows a clear, predictable 24-hour pattern that repeats daily. Network traffic begins to rise around 8-9 AM local time, reaches a peak around 11 AM, drops down a little during lunch time, comes back after 1 PM and shows another peak at around 3-4 PM, then declines by the end of business day around 5 PM. A renewed activity in the early evening hours starts around 8 PM and peaks at 10-11 PM, and diminishes sharply after midnight. This phenomenon was recognized thirty years ago [18][28].

Human activity is the major factor to explain network daily activity patterns. The human daily activity follows a certain path, we can use a deterministic function to describe it. However, for different regions or times, the diurnal patterns may vary, the deterministic function may also be different. The causes of the variation are complex. Network topologies, Internet protocols, Internet service charge policies and other factors may affect patterns.

As shown in Figure 3.2 [18], for the same regional trunk link in the east of US, south

direction and north direction traffic have some differences. To simulating diurnal traffic model, some assumptions need to be made.

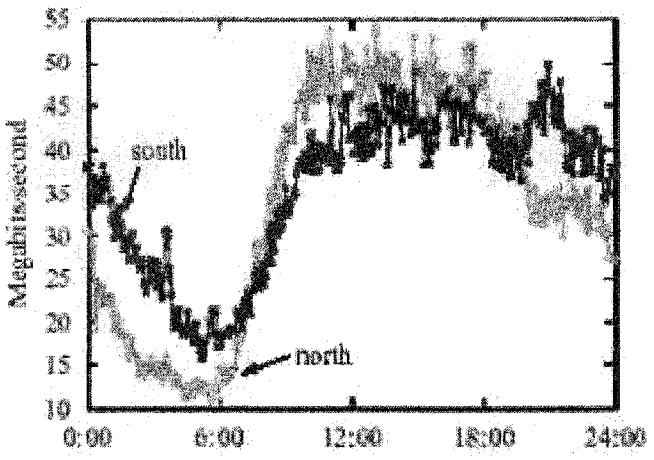

Figure 3.2: Byte Volume for 1 Day on US Domestic Link

Actually, a deterministic function $h(t)$ and a stochastic self-similar process can coexist. Depending on different time scales, one of them or both may become the dominant factors of the traffic models. Brownian motion dominates the behavior from hundreds of milliseconds to minutes level. At tens of minutes and more, diurnal variation is the dominant factor [24].

Two OC-3 trunks of MCI's commercial Internet backbone were measured over two time ranges (24-hour and 7-day) up to 240,000 flows [18]. The time interval between 5 AM and 11 AM was observed that had the largest rate change in the daily pattern; It increased 300-500% . Even in this period of time, traffic rate changes less than 1% in any 4-second interval, the random part dominates the change of the traffic rate.

In this thesis, we simulate daily traffic patten based on the MCI US domestic link real traffic pattern (north bound). We use Matlab curve fitting function to determine the pattern (see Figure 3.3). Since the out-going channel capacity is less than 7.5 Mbps in this thesis, we take smaller incoming traffic rates other than those rates observed in [18]. The corresponding deterministic function $h(t)$ is:

$$h(t) = 1.44 \times 10^{-39}t^8 + 5.0 \times 10^{-34}t^7 - 7.0 \times 10^{-29}t^6 + 5.3 \times 10^{-24}t^5 - 2.2 \times 10^{-19}t^4$$

$$+5.2 \times 10^{-15}t^3 - 6.0 \times 10^{-11}t^2 + 3.2 \times 10^{-7}t - 0.00065$$

where t (unit second) is the time in a day.

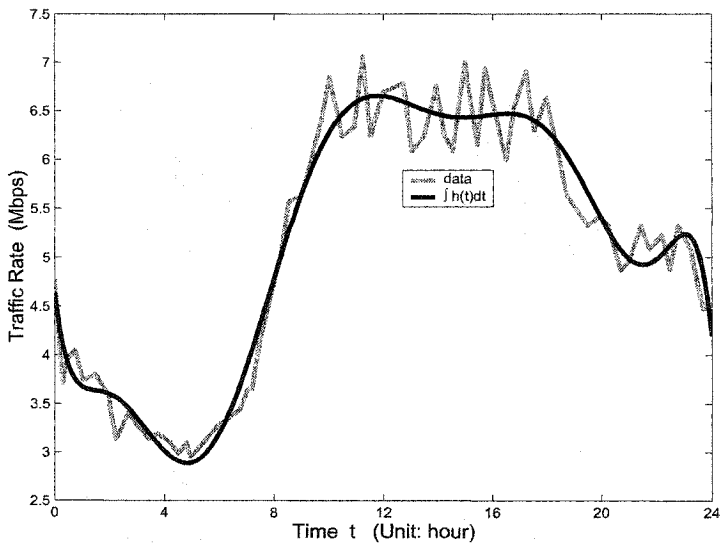

Figure 3.3: Drift Function of US Domestic Link Daily Pattern

Evaluation of Diffusion Parameter

The diffusion function $\sigma(t)$, $t \geq 0$ in (3.2) and (3.3) determines the volatility of the incoming traffic. As we discussed above, in different time scales, one of diffusion and drift components may become dominant factor. The $\sigma(t)$ plays different roles in different time

scales.

For tens of minutes or more time scales, diurnal variation becomes a dominant factor. The study of [18] showed the existing of the diurnal pattern. Traces of real traffic were studied in two ways [41]:

1. by observing how variance changes over the large range of mean bandwidths presented in 24 hours traces;

2. by observing the relationship of variance and mean bandwidth for individual users and combinations of users.

It is found that for the time range as 24 hours, the variance of aggregated traffic changed linearly with the mean of traffic (Figure 3.4). According to the above conclusion, we may consider that the variance of the traffic is estimated by the square root of the traffic rate $a(t)$ with a parameter r on the basis of real traffic volatility

$$\sigma(t) = r\sqrt{a(t)},$$

Thus, the traffic model (3.2) can be rewritten as:

$$\begin{aligned} da_i = \quad & h_i(t)\{I(a_i(t) > 0, h_i(t) \leq 0) + I(h_i(t) > 0)\}dt \\ & +r\sqrt{a(t)}I\{a_i(t) > 0\}dW_i(t), \quad i = 1, 2, \cdots, m \end{aligned}$$

The traffic model (3.3) can be presented in the same way.

For a time scale range from hundreds of milliseconds to tens of minutes, the traffic shows self-similar and LRD characteristics. As we mentioned, in such a short period, $h(t)$ makes little contribution to the traffic variation. Brownian motion dominates the traffic volatility at this time scale.

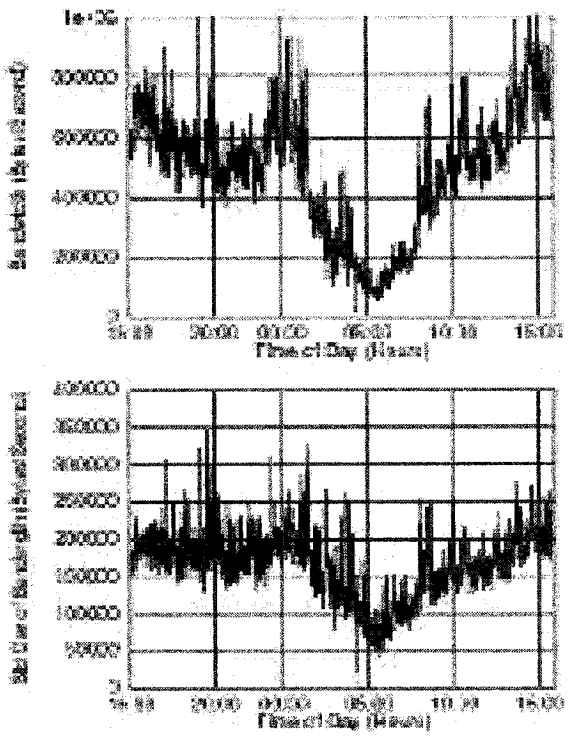

Figure 3.4: Variance of Aggregated Traffic vs. Traffic

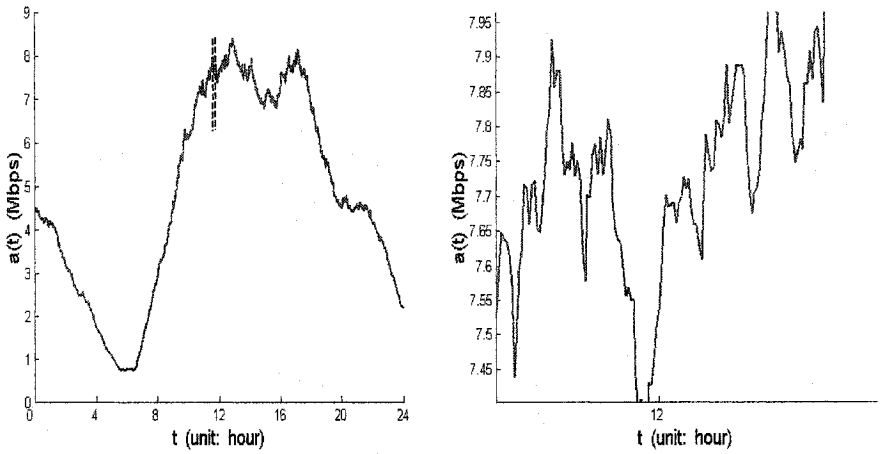

Figure 3.5: The 24-hour Traffic

In Figure 3.5, the part between two dashed lines of the first figure is zoomed in as the second figure. Shown as the figures, the 24-hour traffic, which is generated based on the MCI daily traffic pattern, can capture the diurnal traffic trend, and it shows self similarity and LRD properties in a short period of observing time. Our traffic models can simulate traffic behaviors of the time scale from several seconds to 24 hours. The variance of real traffic has a large range. We can simulate it by adjusting the parameter r in the diffusion component.

3.2 System Model

Considering m users, the state of the system is described by the occupancy status of all the TBs and the multiplexor given by $m+1$ equations. Let $T_i > 0$, $i = 1, 2, \cdots, m$, denote the size of the i -th TB, and $\rho_i(t)$ the number of tokens available at time t in the i-th TB. We can use the following equation to describe the rate of change of the token population, as presented in [2],

$$d\rho_i = \{u_i(t)I(\rho_i(t) < T_i) - a_i(t)I(\rho_i(t) > 0)\}dt, \quad i = 1, 2, \cdots, m \tag{3.5}$$

This is simply a balance equation where the rate of change of token population (count) is given by the algebraic sum of fresh token supply rate u_i (provided the bucket is not full) and the consumption rate a_i due to the arrival of traffic from the source i (subject to the condition that the corresponding bucket is not empty). Clearly TBs will drop nonconforming packets whenever they are empty.

Let $q(t)$ denote the state of the multiplexor measured in terms of (any suitable unit, such as bytes) space occupied in its buffer which has size $Q > 0$. Let $C(t)$ denote the (out going) link capacity available at time t. The multiplexor injects the traffic into the link at a rate determined by the available capacity. The dynamics of the multiplexor can then be formulated as follows:

$$dq = (-C(t)I\{q(t) > 0\} + \sum_{i=1}^{m} a_i(t)I(\rho_i(t) > 0)I(q(t) < Q))dt \tag{3.6}$$

This equation describes the occupancy status of the multiplexor. The rate of change of the multiplexor queue is determined by the algebraic sum of link service rate (provided the multiplexer is not empty) and the sum of all conforming traffic rates from all the token buckets (subject to the condition that the corresponding multiplexer is not full).

In order to achieve QoS satisfaction of each accessing traffic and maximum utilization of the multiplexor, the controller adjusts the token supply at each TB to control the access

rate of each traffic. This can be used in current networks to improve utilization by dynamically allocating bandwidth to each user [1][2].

Remark. Note that in developing these dynamic equations, we have assumed that both the TBs and the multiplexor are noiseless.

Considering the equations (3.5) and (3.6) and defining

$$f_i(t, \rho_i, a_i, u_i) \equiv \{u_i I(\rho_i < T_i) - a_i I(\rho_i > 0)\} \tag{3.7}$$

$$f(t, \rho, a, u) \equiv column\{f_i(t, \rho_i, a_i, u_i), i = 1, 2 \cdots m\} \tag{3.8}$$

$$g(t, \rho, a, q) \equiv (-C(t) I\{q > 0\} + \sum_{i=1}^{m} a_i I\{\rho_i > 0\} I\{q < Q\}) \tag{3.9}$$

we can rewrite all the equations from (3.7)-(3.9) as a system given by

$$da = b(t, a)dt + \sigma(t, a)dB \tag{3.10}$$

$$d\rho = f(t, \rho, a, u)dt \tag{3.11}$$

$$dq = g(t, \rho, a, q)dt. \tag{3.12}$$

This is a $2m + 1 \equiv n$ dimensional stochastic differential equation describing the dynamics of our system that consists of the source (traffic) dynamics and the dynamics of the access control mechanism at the edge of a backbone network. For analysis of the system, it is more convenient to write these equations as an n-dimensional stochastic differential system. Define the state vector as $\xi \equiv row\{a, \rho, q\}$ and the drift and the diffusion matrices as

$$F \equiv column\{b, f, g\} \quad G \equiv matrix\{\sigma, 0, 0\},$$

where F is a n-vector, G is a $n \times m$ matrix valued functions as indicated, σ is $m \times m$ matrix, the rest elements in matrix G are 0. Using these notations, we can rewrite our system in the compact form

$$d\xi(t) = F(t, \xi(t), u(t))dt + G(t, \xi(t))dB, t \geq 0. \tag{3.13}$$

Chapter 4

Objective Function and Performance Measures

In this chapter, we introduce the objective function, which considers losses at TBs, losses at the multiplexor and the waiting time in the queue of the multiplexor. And then we define several possible measures for evaluating the overall performance of the system. These are system congestion, expected first time to congestion and expected residence time in congestion zone.

4.1 Objective Function

The objective functional for a network provider proposed in [2] is given by:

$$\begin{aligned} J(u) \equiv \quad & E\{\int_l \lambda_1(t)\{\sum_{i=0}^{n} a_i(t) I(\rho_i(t) = 0)\}dt \\ + \quad & \int_l \lambda_2(t)\{(\sum_{i=0}^{n} a_i(t) I(\rho_i(t) > 0)) I(q(t) = Q)\}dt \\ + \quad & \int_l \lambda_3(t) q(t) dt\} \end{aligned} \tag{4.1}$$

The functions $\lambda_i(t),\ i = 1,\ 2,\ 3$ are nonnegative measurable functions used as weights or importance given to each of the losses. This can be chosen by network designers to reflect different concerns and scenarios as necessary. If the amount of incoming traffic is larger than that of tokens in corresponding TBs, the exceed portion traffic will be dropped, which we call TB loss, the rest data will be conformed and sent to multiplexor. The multiplexor throws away conformed traffic after all of its space is occupied. The multiplexor loss occurs at that moment. On the right hand side of the equation (4.1), the first term represents (traffic or cell) losses at TB, the second term gives the losses at the multiplexor, and the third term is a measure of the service delay. Note that these are implicit functions of the control policy. Clearly, in terms of our final state equation given by (3.13), the cost functional (4.1) can be compactly written as

$$J(u) = E\Big\{\int_0^T \ell(t, \xi(t))dt\Big\} \tag{4.2}$$

where ℓ denotes the integrand of the expression (4.1). One of the objectives of network service providers is to improve the system performance by using control strategies that minimize this cost functional. Note that control policy has no influence on the source, though this information must be used by the controller to achieve its goal leading to feedback control.

Later we will discuss how different feedback and open loop control policies influence the objective functional.

4.2 Other Performance Measures

In addition to the concern for cell losses in the system, there are other measures of the system performance which are equally important. For example, the first time congestion occurs, and residence time in the state of congestion. Here we define several related concepts.

4.2.1 System Congestion

Congestion is a state of the network resources in which the aggregate demand may exceed the capacity of the system. The multiplexor is close to its full capacity or demand for bandwidth exceeding the available link capacity during certain periods of time. Clearly, it is expected that congestion may result in degradation of service quality, QoS, to the end users [37]. For the system under consideration, we may define the congestion as follows.

The system is regarded to be in the state of congestion when the multiplexor state hits the zone (closed interval) $Q_\alpha \equiv [\alpha Q, Q]$, where $0 < \alpha < 1$, and Q is the buffer size of the multiplexor.

In general, one may consider the system to be in the state of congestion if a certain percentage of the buffer space is occupied. The value of α can be chosen arbitrarily by a network provider. In this thesis, we choose $\alpha = 0.90$.

4.2.2 Expected First Time to Congestion

Traffic loads in a network keep changing randomly, with peaks and valleys occurring during certain periods of operation. Considering the traffic observation period to be T, let C_T denote the set of time instants during which the multiplexor hits the congestion zone. In symbols, this is given by,

$$C_T \equiv \{t \in [0, T] : q(t) \in Q_\alpha\} \tag{4.3}$$

where $q(t)$ is the state of the multiplexor. The first time to congestion, denoted by, τ_c, is then given by,

$$\tau_c \equiv \begin{cases} \inf\{C_T\}, & \text{if the set is nonempty} \\ T, & \text{if the set is empty} \end{cases} \tag{4.4}$$

We are interested in the expected values of the equation (4.4), and define the expected first time to congestion as:

$$T_c \equiv E(\tau_c) \tag{4.5}$$

4.2.3 Expected Residence Time in Congestion Zone

Residence time in congestion zone is the total time spent by the system in the state of congestion. For any sample path $q(t), t \in T$, we denote this by τ_r as follows:

$$\tau_r \equiv \int_0^T I\{q(t) \in Q_\alpha\} dt \tag{4.6}$$

Just as the way we define (4.5), the expected value of the equation (4.6) is given as follows,

$$T_r \equiv E(\tau_r) \tag{4.7}$$

Note that the cost functional given by (4.6) can be absorbed in the general functional given by (4.2) by adding the integrand of (4.6) to those of (4.1) with desired weight. Later in the sequel we will use Monte Carlo techniques to compute this functionals and the measures of performance.

Chapter 5

Control Strategies

It is desirable that all incoming traffic is served right away; However, the resources are limited. Heavy traffic may lead to congestion, which is one of the most serious problems in network operation and management. An Internet service provider (ISP) has to apply a network-access bandwidth policy to ensure that traffic falling within specified rate parameters is transmitted, while traffic exceeding the acceptable limits is dropped [11].

To find the optimal control for the dynamic system (3.13) with the cost functional (4.2), one may consider to make use of dynamic programming [43]. In case B (B represents either the standard Brownian motion or the FBM in the equation (3.13)) is a standard Brownian motion, with values in R^m, following Bellmans principle of optimality, one can derive the so called HJB (Hamilton-Jacobi-Bellman) equation. Unfortunately, the equation is a degenerate equation with discontinuous coefficients. These problems have not been considered in the literature so far. In case B is a Fractional Brownian motion, the problem is more complex than that of the standard Brownian motion.

In view of these theoretical difficulties, we choose not to find the absolute optimum, instead, look for simple and practical but sufficiently efficient control strategies. We

propose two feedback control strategies: one is a simple feedback control law based on engineering intuition and judgement, another is a feedback control law based on neural network. Before we propose those feedback controls, we discuss some standard open loop controls used in current network management practice. We compare the performance of the feedback policies with those corresponding to open loop strategies. Note the loops that we consider in this thesis are the links between TBs and multiplexor outgoing channel.

5.1 Open Loop Control

As it is used on Cisco edge routers [11], open loop control applies a constant token generation rate. The token generating rate is chosen based on traffic statistical properties such as the mean rate, peak rate and variance. The advantage of open loop control is very simple to implement in practice. However, since incoming traffic rates, the token population in TBs, and the state of multiplexor change dynamically with time, this control strategy do not consider the variation of the system availability, the system cannot adapt to the change. As a result, it may either trap in the congestion state frequently or waste valuable resources while heavy traffic arrives.

In our numerical experiments, we give identical token generation rates to different TBs in the same system, and take mean and peak traffic rates as token generation rates. The cost functional, and other performance related factors are computed respectively.

5.2 Feedback Control

Considering the weakness of open loop control, one must seek for a tradeoff between reducing congestion (occurring) probability and increasing traffic throughput. In this section, we discuss our feedback control methods.

5.2.1 Simple Feedback Control Law

Simple Feedback Control Law (FBC)

The state of multiplexor is very important regarding the current ability of the system. We propose a control law that considers the available space of the multiplexor and its rate of change with time. The control law is given as follows:

$$u_i(t) \equiv \left(\frac{a_i(t)}{\sum_{j=1}^{m} a_j(t)} \right) \{ F_i(\dot{q}(t)) + k_{i,3} e^{\beta_i (Q - q(t))} I(0 \leq q(t) < Q) \} \tag{5.1}$$

where the function F_i is given by

$$F_i(z) = \begin{cases} -k_{i,1} z & z < 0 \\ k_{i,2} e^{-\lambda_i z} & z > 0. \end{cases} \tag{5.2}$$

The constants $\{k_{i,1}, k_{i,2}, k_{i,3}, \lambda_i, \beta_i\}$ are nonnegative parameters to be chosen for performance improvement. The control law, representing the token generation rate, as proposed above, is proportional to the fractional demand of the users multiplied by factors which take into account the rate of change of the multiplexor space and the available space. This is a simple control law that can be physically implemented based on measured traffic data and observed multiplexor status, rate and the state.

In order to avoid monopoly by any of the users, it is necessary to include an extra factor in the above control policy changing the fraction

$$\left(\frac{a_i(t)}{\sum_{j=1}^{m} a_j(t)} \right) \Longrightarrow \left(\frac{a_i(t)}{\sum_{j=1}^{m} a_j(t)} \right) \wedge (\mu_i / m) \tag{5.3}$$

where $\mu_i \in N_m \equiv \{1, 2, \cdots, m\}$. Clearly, if $\mu_i = r$, $r \in N_m$, the i-th user cannot exceed the fraction r/m, thereby the network management can exercise control on the distribution of resources, avoid monopoly and even offer priorities. In fact, one can choose μ_i as

a function of time.

In chapter 8, the performance of this feedback control law is evaluated systematically.

Feedback Control Parameters Optimization

In our system model, for a given but arbitrary traffic process, the five parameters (k_1, k_2, k_3, λ, β) in the feedback control law determine the value of the cost function (4.1). Our aim is to find a set of parameters that can lead to the minimum cost.

Nonlinear optimization methods have been studied for hundreds of years, and a huge amount of literature discussed this subject in fields such as operations research, numerical analysis, and statistical computing. So far no single method is the best for all nonlinear optimization. The method chosen to solve a problem should depend on the characteristics of the problem. For functions with continuous second derivatives, there are three general types of algorithms: stabilized Newton and Gauss-Newton algorithms, various Levenberg-Marquardt and trust-region algorithms, and various quasi-Newton algorithms. For most practical purposes those methods have been found to be effective. However, all of the above methods search local optima [44].

We take advantage of medium-scale algorithms in the Matlab Optimization Toolbox, which implements Gauss-Newton Levenberg-Marquardt, and trust-regionmethod algorithms, to minimize the cost functional and obtain an optimal set of parameters. Due to indicators used in the system model, the figure in the 6-dimension (the five parameters and the cost) space is quite rough. The minimized object function usually is a local optimal value closest to the starting guess of the input parameters. Moreover, it is also a time consuming task. For our problem, this method is not practical.

A desired method should be the one that can give global minimum and deal with stochas-

tic traffic. There are a variety of approaches to find global optimization. A simple way is to run any of the local optimization methods from numerous random starting points, or one can apply more complicated methods designed for global optimization such as simulated annealing (SA) or genetic algorithms [44].

Simulated annealing, as we introduced in section 2.9, accepts lower energy state and also randomly updates new state with higher energy state. This algorithm searches for the absolute minimum of the objective function over the complete range of input parameters, and trends to find, under certain condition, the global minimum and not get stuck in a local minima, even in the presence of many local minima [39]. It is a probabilistic optimization technique well suited to solve our problem.

To implement this algorithm, we take the following steps:

1. initialize the starting temperature T, a positive integer m_0 which is the number of times reducing the temperature T, a positive integer n_0 which is the number of iterations at each temperature, and the cost functional J_p;

2. set $n = 0$, where n is the number of iterations at each temperature level;

3. generate a set of positive random numbers (k_1, k_2, k_3,λ, β);

4. substitute these five control parameters into the equations (5.1) and (5.2), calculate the token generating rate $u_i(\ i = 1,\ ,\ 2,\ \cdots,\ m)$ of the ith token bucket, and then compute the cost function (4.1) with u_i as controls, the result, denoted by J_c, is the current cost.

5. calculate $\Delta J = J_p - J_c$;

6. in case of $\Delta J < 0$, then update $J_p = J_c$; in case of $\Delta J \geq 0$, generate an uniformly distributed random number $p \in (0, 1)$: if $exp(-\frac{\Delta J}{kT}) > p$, update $J_p = J_c$, otherwise,

keep J_p as it is;

7. check n, if $n = n_0$, go to the step 8, otherwise let $n = n + 1$, go back to step 3;

8. check m, if $m = m_0$, end the procedure, otherwise let $m = m + 1$, decrease T based on a certain rule, go to the step 2.

The final J_p is treated as the lowest cost, and the corresponding set of (k_1, k_2, k_3,λ, β) is the optimal set of feedback control parameters. The flow chart of this procedure is illustrated in Figure 5.1.

There are several aspects that have to be considered:

- The ranges of control law parameters;
- The number of iterations at each temperature;
- Temperature decrement.

As we defined in (5.1), (5.2), the ranges of the five constraints are given as following: $k_1 \geq 0$, $k_2 \geq 0$, $k_3 \geq 0$, $\lambda \geq 0$, $\beta \geq 0$. Upper bounds are determined by experiences.

Theoretically, one should allow enough iterations at each temperature in order to stabilize the system at that temperature. Unfortunately, to achieve this, the number of iterations at each temperature might increase with problem size exponentially. In most real situations it is satisfactory to find a solution yielding a near-optimal return [45]. As a compromise, we can either use a large number of iterations at a few temperatures, a small number of iterations at many temperatures or a balance between the two.

There are two ways to decrease the temperature: simple linear decrement, and geometric decrement (next temperature t_n is based on current temperature t_c, $t_n = \alpha t_c$, where

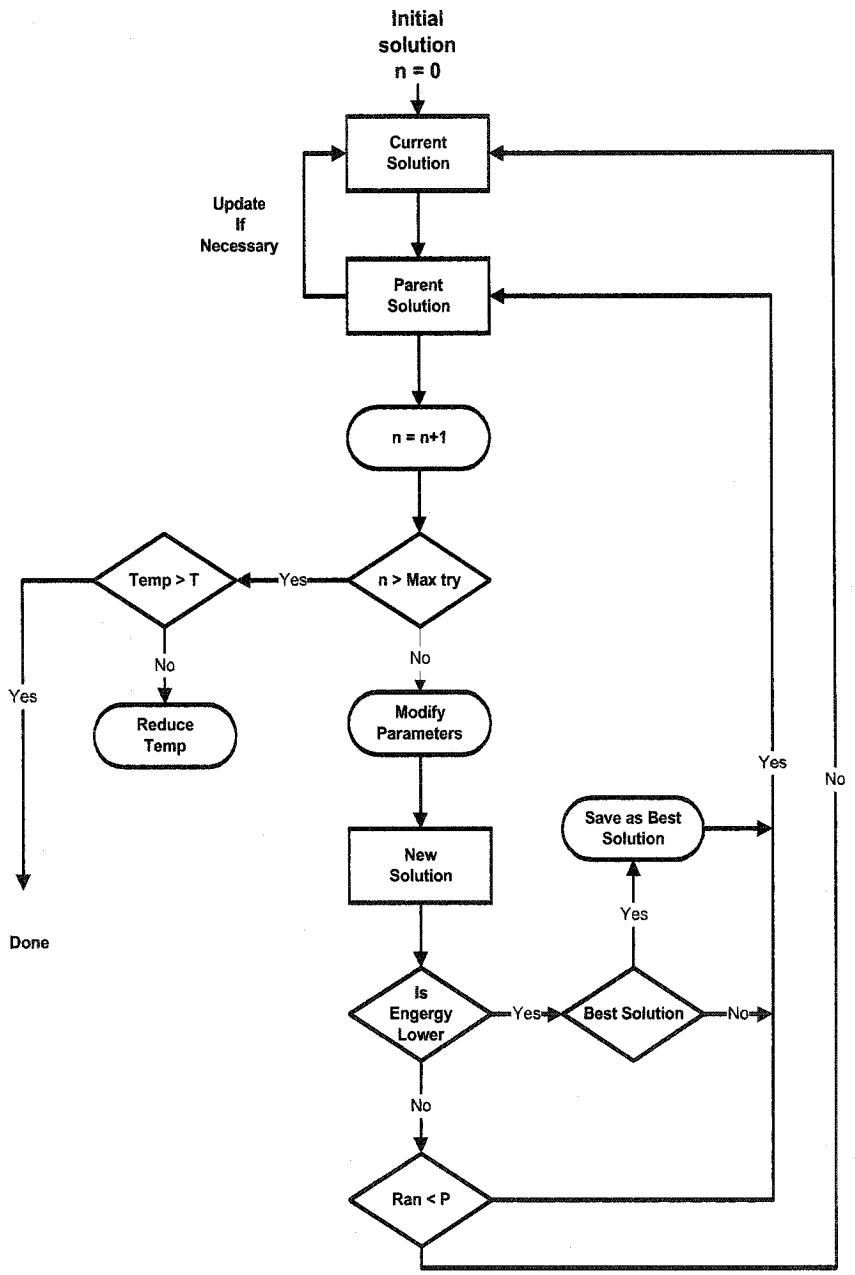

Figure 5.1: Simulated Annealing Algorithm Flow Chart

$\alpha < 1$). In this thesis, we use the second method and take $\alpha = 0.6$.

For convenience, we take $k = 1$. Concerning the initial temperature, it could be a very large number, but according to our experience, starting from 20 can reach to a similar result as from higher temperature. As to the number of iterations at each temperature, most papers suggest taking a constant number of iterations for each temperature.

5.2.2 Feedback Control Based on Neural Network

A neural network (see more details about neural network in Section 2.10) is a flexible mathematical structure which is capable of identifying complex nonlinear relationships between input and output data sets. Neural network models have been found useful and efficient, particularly in problems for which the characteristics of the processes are difficult to describe by using mathematical equations. In theory, a neural network of two layers can be trained to approximate any function (with a finite number of discontinuities) arbitrarily well if there is enough neurons used [46]. For our problem, the advantage of neural network makes it a desired tool to approximate the control function instead of solving complex methodologies such as HJB equations.

We consider a two layers neural network shown as Figure 5.2. ℓ ($\ell = 2m + 1$) inputs $p_1, p_2, \ldots, p_\ell$ represent incoming traffic rates ($a_1, a_2, \cdots, a_m$), TBs population ($\rho_1, \rho_2, \cdots, \rho_m$) and multiplexor state q respectively. We put n neurons in the hidden layer, and m neurons in the output layer. The outputs are m token generating rates $u_1, u_2, \cdots, u_m$ for token bucks 1, 2, $\cdots$, m respectively. Let R_{1j} denote the output of the j th neuron of the first layer.

$$R_{1j} = f_1(\sum_{i=1}^{\ell} W_{ij}^I p_i + b_j^1), \quad j = 1, 2, \cdots, n \tag{5.4}$$

where W_{ij}^I is the weight given to the connection between the i th input and the j th

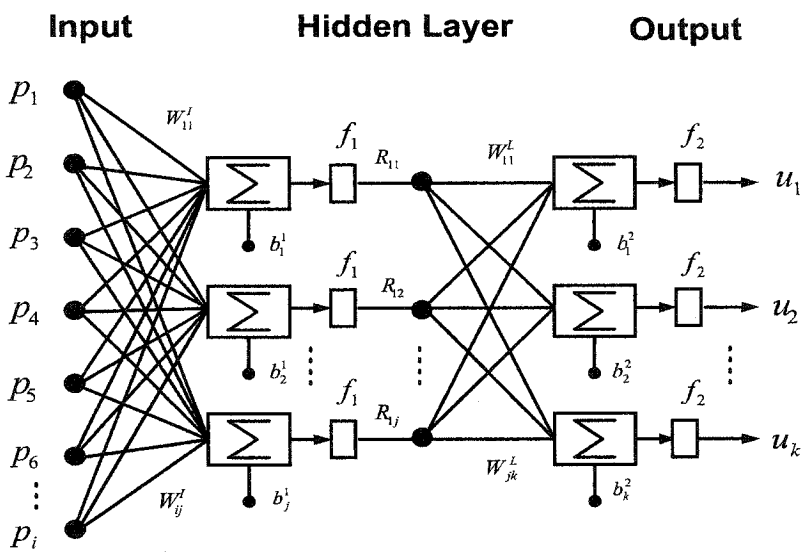

Figure 5.2: Neural Network Structure

hidden layer neuron. b_j^1 represents the bias of the j th neuron in the first layer. f_1 is the transfer function of this layer.

The neural network outputs u_k can be written as follows,

$$\begin{aligned} u_k &= f_2(\sum_{k=1}^{m} W_{jk}^L R_{1i} + b_k^2) \\ &= f_2(\sum_{k=1}^{m} W_{jk}^L f_1(\sum_{j=1}^{\ell} W_{ij}^I p_i + b_j^1) + b_k^2), \quad k = 1,\ 2, \cdots,\ m. \end{aligned} \tag{5.5}$$

where W_{jk}^L is the weight given to the connection between the j th output of hidden layer and the k th output layer neuron. b_k^2 represents the bias of the k th neuron in the output layer. f_2 is transfer function of this layer. The weight vector W is given by $W \equiv \{W_{ij}^I,\ W_{jk}^L,\ b_j^1,\ b_k^2,\ i = 1, 2, \cdots, \ell;\ j = 1, 2, \cdots, n;\ k = 1, 2, \cdots, m\}$. All these can be compactly described by the following weighted nonlinear transformation

$$u \equiv N(W, x) \tag{5.6}$$

where $u_r = N_r(W, x),\ r = 1, 2, \cdots, m$. Identifying x with the state vector $(a, \rho, q)'$ and substituting the control law given by equation (5.6) into the state equation (3.13), we arrive at the closed loop system given by:

$$d\xi(t) = F(t, \xi(t), N(W, \xi(t))dt + G(t, \xi(t))dB, t \geq 0. \tag{5.7}$$

The corresponding cost functional (4.2) may now be written as a function of the weight vector W,

$$J(W) \equiv E \int_0^T \ell(t, \xi(W, t))dt,$$

where now the state depends obviously on the choice of W. This is the functional to be minimized by an appropriate choice of the weight.

To minimize the objective functional and obtain an optimal set of weights and biases, typically, one may compare current outputs with the outputs that are already known, adjust the weights and biases based on the errors. For the theoretical difficulties, the minimal costs are unknown, we do not have proper data to train the neural network. Again, we use simulated annealing algorithm to find the the lowest cost and corresponding weight vector W.

We apply the obtained weight W in the neural network to control token generating rates. The numerical results shows that the cost is further reduced by using this method.

Chapter 6

Numerical Methods

In order to implement continuous mathematical models, numerical methods have to be applied. We use the Euler-Maruyama (EM) method to approximate a stochastic differential equation (SDE). Since some unreasonable negative values may be generated during the simulation, an adaptive step method is proposed to avoid this problem from happening. In this chapter, we introduce the Euler-Maruyama method and an adaptive time interval step method.

6.1 Euler-Maruyama Method

To simulate network traffic, our traffic models take an Ito version of SDE, the SDE can be be rewritten in the differential form as [48]:

$$dX(t) = f(X(t))dt + g(X(t))dW(t), \quad X(0) = X_0, \quad 0 \leq t \leq T \tag{6.1}$$

where f and g are scalar functions and the initial condition X_0 is a random variable. The second integral on the right hand side of (6.1) is the Brownian motion. For each t, $X(t)$ is a random variable.

It can also be rewritten in an integral equation form as follows:

$$X(t) = X_0 + \int_0^t f(X(s))ds + \int_0^t g(X(s))dW(s), \quad 0 \le t \le T \tag{6.2}$$

$$\int_{t_{j-1}}^{t_j} dx = \int_{t_{j-1}}^{t_j} f(X(t))dt + \int_{t_{j-1}}^{t_j} g(X(t)dW(t)$$

To apply a numerical method to (6.2) over $[0, T]$, the interval has to be discretized first. Let $\Delta t = T/L$ for some positive integer L.

$$\int_{t_{j-1}}^{t_j} f(X(t))dt = x_j - x_{j-1}$$

The second and third items of (6.2) can be approximated as

$$\int_{t_{j-1}}^{t_j} f(X(t))dt \approx f(X_{j-1})\Delta t$$

$$\int_{t_{j-1}}^{t_j} g(X(t)dW(t) \approx g(X_{j-1})(W_j - W_{j-1})$$

Thus (6.2) can be written as the form:

$$X_j \approx X_{j-1} + f(X_{j-1})\Delta t + g(X_{j-1})(W_j - W_{j-1}), \quad j = 1, 2, \cdots, L \tag{6.3}$$

This is known as the Euler-Maruyama method for solving stochastic differential equations.

With the EM method, our continuous traffic simulation model that is based on standard Brownian motion can be implemented as the following:

$$\begin{aligned} a(t + \Delta t) = \quad & a(t) + h(t)\{I(a(t) > 0, h(t) \le 0) + I(h(t) > 0)\}\Delta t + \\ & r\sqrt{a(t)}I(a(t) > 0)R_g\sqrt{\Delta t} \end{aligned} \tag{6.4}$$

where R_g is Gaussian random number with N(0, 1).

In case of fractional Brownian motion, it takes the form of

$$X_j \approx X_{j-1} + f(X_{j-1})(t_j - t_{j-1}) + g(X_{j-1})(B_j^H - B_{j-1}^H), \quad j = 1, 2, \cdots, L. \tag{6.5}$$

The traffic simulation model that is based on fractional Brownian motion can be written as follows,

$$\begin{aligned} a(t+\Delta t) = \quad & a(t) + h(t)\{I(a(t) > 0, h(t) \leq 0) + I(h(t) > 0)\}\Delta t + \\ & r\sqrt{a(t)}I(a(t) > 0)(B^H_{t+\Delta t} - B^H_t) \end{aligned} \tag{6.6}$$

For numerical computation of the above equation, $B^H_j - B^H_{j-1}$ can be approximated by

$$\begin{aligned} B^H_j - B^H_{j-1} \approx \quad & K_H(t_j - t_{j-1})(W(t_j) - W(t_{j-1})) + \\ & \sum_{i=1}^{j-1}(K_H(t_i - t_r) - K_H(t_{i-1} - t_r))(W(t_i) - W(t_r)) \end{aligned} \tag{6.7}$$

where t_r can be any value that is smaller than t_i and t_{i-1}, and $W(t_j) - W(t_{j-1})$ and $W(t_i) - W(t_r)$ are independent Gaussian random numbers [25]. K_H is a kernel function in the equation (6.7). For our numerical experiments, we choose the simple kernel K_H given by

$$K_H(t) = C_H t^{(H-\frac{1}{2})}, \quad \frac{1}{2} < H < 1,$$

where H is a Hurst parameter, and C_H is any constant.

6.2 Implementation with Adaptive Step

As the EM method is applied to approximate the SDE, no matter how small the fixed time interval Δt is, the system may generate a negative traffic rate, TB population, or multiplexor state values. In real networks, all of them should be non-negative values, those negative values obviously make no physical sense.

For instance, in the equation (6.4), the traffic rate $a(t)$ is a non-negative value. The deterministic function $h(t)$ and the random number R_g can be negative numbers. This may result in negative $a(t+\Delta t)$ by (6.4). Therefore we have to find a method to overcome this problem. One solution is to take a smaller time interval, it can reduce the probability of

generating negative values, but cannot prevent it from occurring. Moreover, the smaller the time intervals are, the longer computing time is required.

There are some better techniques to deal with this problem. For example, one can solve it in a traditional manner by adding an extra item on the right hand side of the equation (6.4) (see [47]). This item increases whenever other items on the right hand side try to generate a negative value. The added item keeps the total value positive. However, how to choose the value of this item may raise another problem.

We propose a new method, which applies adaptive time intervals, to avoid the generation of a negative value. On the right side of (6.4), the first item is the pervious time interval traffic rate, which is not related to Δt. The sum of the second and third items on the right side of (6.4) is the rate change during the current time interval, which is a function of Δt. We give the system a relatively large time interval at beginning. If the traffic rate decreases very quickly, the system may generate a negative value. Whenever $a(t + \Delta t)$ is found to be negative, we keep cutting the time interval and test the results until a non-negative $a(t + \Delta t)$ obtained. The time interval regains its initial step whenever possible. With this step adjusting technique, programs can run fast and avoid negative results. We implemented this method in all of our Matlab simulation programs.

To illustrate this method, we use the traffic model (6.4), compare the results of which the method is applied before and after. In Figure 6.1, with a fixed step time interval, negative traffic rates occur in 178 time intervals. After we apply the above method (illustrated as in Figure 6.2), with the same set of system parameters, the system does not generate any negative values.

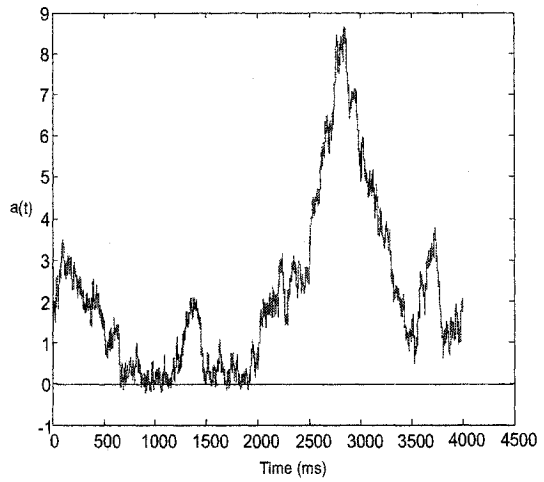

Figure 6.1: Traffic Generated with Fixed Step

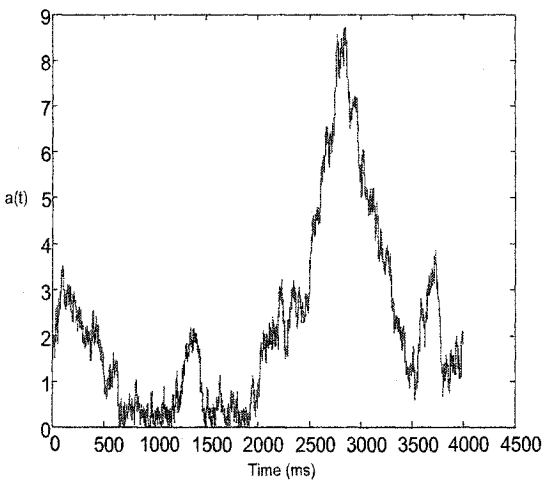

Figure 6.2: Traffic Generated with Adaptive Step

Chapter 7

Basic Data Used for Simulation Experiments

For the numerical simulation, we assume the system has a simple scenario comprised of three traffic sources policed by three token buckets with their (conforming) outputs multiplexed by a single multiplexor.

7.1 System Parameters Used for Simulation

In all the numerical experiments, we make two assumptions. First, the traffic generated by the three independent users have the same statistical characteristics and a similar traffic mean rate and variance. Packet size is not taken into consideration. Second, there is no feedback control delay. Depending on the requirements of different applications, the weights of TB losses, the multiplexor loss, and the delay loss in the objective function can be taken differently. For some applications, it is preferable to drop packets at token buckets rather than at the multiplexor. In our experiments, we give the greatest weight to the loss at multiplexor, the loss at TB gets more weight than the delay loss. We choose

$\lambda_1(t) = 5$, $\lambda_2(t) = 10$ and $\lambda_3(t) = 0.1$ for all i. The system configuration and parameters are given in Table 7.1; here Q denotes the buffer size of the multiplexor, K represents the number of time intervals in a trace, and M is the number of sample paths. State initialization is set as: $\rho_i(t_0) = 0,\ i = 1, 2, 3;\ q(t_0) = 0$.

TB Size	Q	Δt	K	M
1500 Bytes	4000 Bytes	0.001 Sec	4000	300

Table 7.1: System Configuration and Parameters

The structure of the neural network that is used for our experiment is shown in Table 7.2.

Number of Inputs	7
Number of Outputs	3
Number of Layers	2
Number of Neurons in Layer 1	3
Number of Neurons in Layer 2	3
Transfer Function in Layer 1	tangent sigmoid
Transfer Function in Layer 2	positive linear

Table 7.2: Neural Network Parameters

7.2 Specification of Traffic Traces

In the experiment, we use the FBM traffic model (6.6) to generate different 4-second (traffic) traces as our traffic sources. Based on the Bellcore real traffic we calculated, the Hurst parameter is taken as 0.7625.

As we discussed in Section 6.1, $W(t_j) - W(t_{j-1})$ and $W(t_i) - W(t_r)$ in the equation (6.7)

are Gaussian random numbers. Theoretically, traffic traces in networks may be identical, but the probability of this occurrence is very very small. So we need to avoid identical random numbers. To generate these data, a source of random numbers is required. It is impractical for a computer to produce truly random numbers. Thus, so called pseudo-random number generators have been developed, and these can perform very effectively. We make use of the MATLAB pseudo-random number generator *randn* to produce normally distributed numbers with a mean of 0 and a standard deviation of 1. Since the seed can be any integer between 1 and 2^{31}, for our experiments, there is almost no chance to repeat a same pseudo-random number.

Since we use the Monte Carlo method to calculate expected values, the number of sample paths required in our experiments has to be considered. To determine whether the expected values have converge or not as increasing the number of sample paths, one must have some criterion. We consider that the expected values are convergence, if as the increment of the number of sample paths, the cost variance is less than 1%. Actually, as illustrated in Figure 7.1, this number strongly depends on the variance of incoming traffic, the larger the variance is, the more sample paths are required. In our experiments, we take $r = 0.1$ of the equation (6.4), Figure 7.2 shows that 300 traces of samples is sufficient for the traffic we generate.

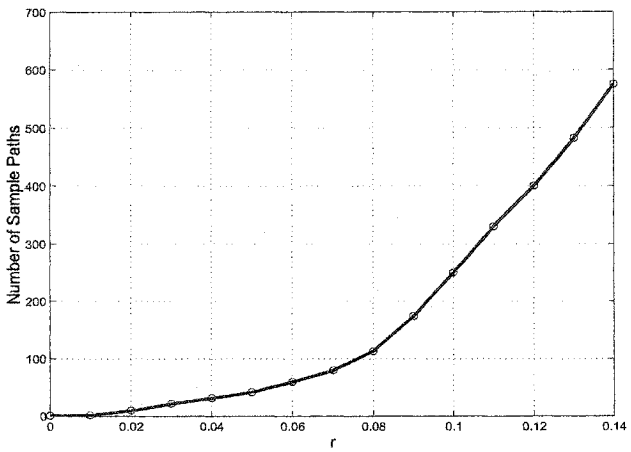

Figure 7.1: Number of Sample Paths vs. Variance

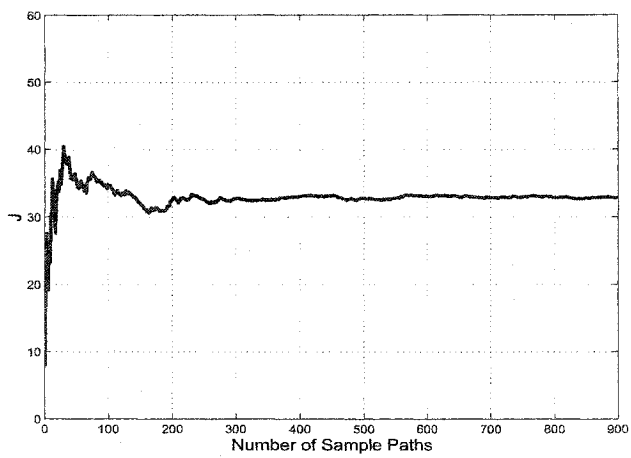

Figure 7.2: Costs vs. Number of Sample Paths

Chapter 8

Numerical Results

In this chapter, we present and evaluate numerical results. The chapter begins by comparing performances of different controls while the channel capacity is a constant. Then we study system performances under variable channel capacity. The experience of the selection on simple feedback control parameters is depicted in the section 4.3. In the last section, we discuss the feedback control based on neural network.

8.1 Dependence of Performance on Control Polices

In Figure 8.1, the costs of FBC, OPM and OPP are plotted as functions of channel capacity. In Figure 8.2, the expected first hitting times (of the congestion zone) corresponding to FBC, OPM and OPP are plotted against channel capacity.

Comparison of Costs: As expected, the costs corresponding to feedback and open loop controls decrease with the increase of channel capacity. At TBs, The OPM strategy drops some packets that the system actually could transfer, while the OPP strategy does not. It can let TBs conform as much traffic as it can. Thus OPP performs better than OPM in terms of cost while the the feedback strategy FBC performances is the best.

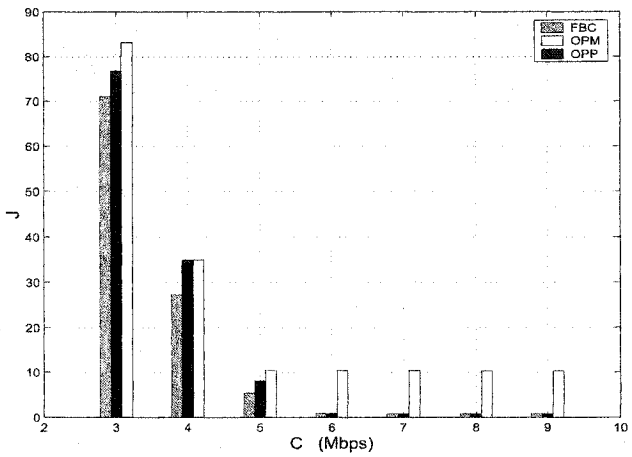

Figure 8.1: Costs vs. Constant Channel Capacity

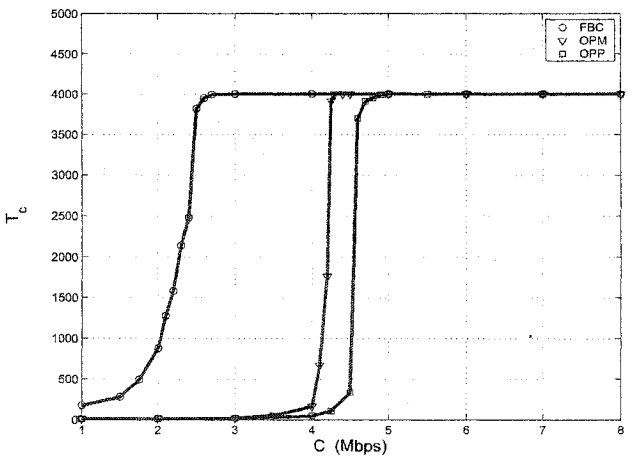

Figure 8.2: First Time to Congestion vs. Channel Capacity

Comparison of Congestion Times: Since the mean (incoming) traffic rate is 4.2638 Mbps in our simulation, the control strategies OPM and OPP develop congestion much earlier at channel capacity lower than 3 Mbps (See Fig.8.2). Obviously, the token generation rate is too large for the system to handle. As channel capacity is increased to 6 Mbps or beyond, the system can certainly transfer the conformed traffic without causing congestion in the multiplexor. For channel capacity close to the mean traffic rate, the peak rate strategy OPP causes congestion earlier than the OPM strategy, while the feedback policy FBC maintains better performance even at channel capacities much lower than the mean traffic rate.

Compared with open loop policies, the feedback control strategy (FBC) behaves better both in terms of cost and congestion. Adjusting token generating rates according to available multiplexor space and its change rate is the reason of the better performance. In brief, the feedback control policy (FBC) offers the lowest cost and much less congestion.

8.2 Dependence of Performance on Channel Capacity

In Section 8.1, the feedback control policy displayed performance better than open loop policies do for a fixed channel capacity. Here we discuss the response of FBC, OPM and OPP to a variable channel capacity. Since any periodic signal can be represented by a sum of sinusoids, we assume that the channel capacity follows a sinusoidal function like:

$$c(t) = c + Ac\sin(2\pi ft/4000)$$

where c is a constant (5 Mbps in the thesis), A is the amplitude of the sinusoid, and f is its frequency. Figure 8.3 (where $A = 0.3$, $f = 1$) shows that the variable capacity,

$c(t)$, increases the costs of all three corresponding control policies, though the FBC policy keeps the lowest cost under the same conditions.

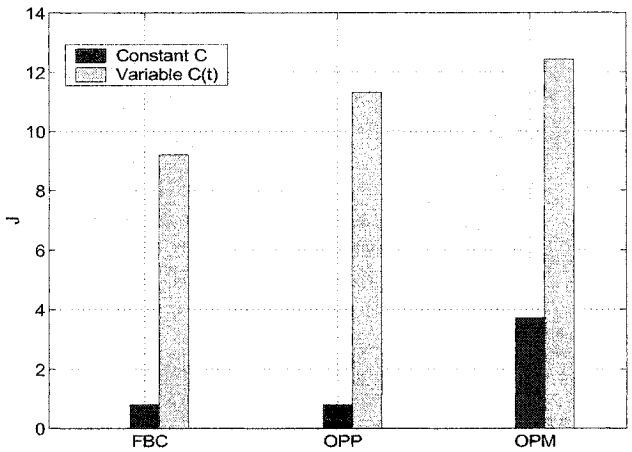

Figure 8.3: Costs for Different Control Strategies

The variation of A and the frequency f are two important parameters of the variable $c(t)$. It is natural to evaluate the influence of these two factors on the system performance.

Dependence of Performance on Variation of Channel Capacity

As shown in Figure 8.4, the cost increases with the increase of the channel parameter A, which determines the magnitude of channel variation. The feedback policy, FBC, performs better than both OPM and OPP in the whole range of variations, except for $A = 0$ where FBC and OPP have the same values. This is due to the fact that the system has sufficient capacity ($c = 5$Mbps) to transfer all incoming traffic (mean rate 4.2638 Mbps) under that condition.

The proportion of time that the channel capacity is less than incoming traffic rate is cru-

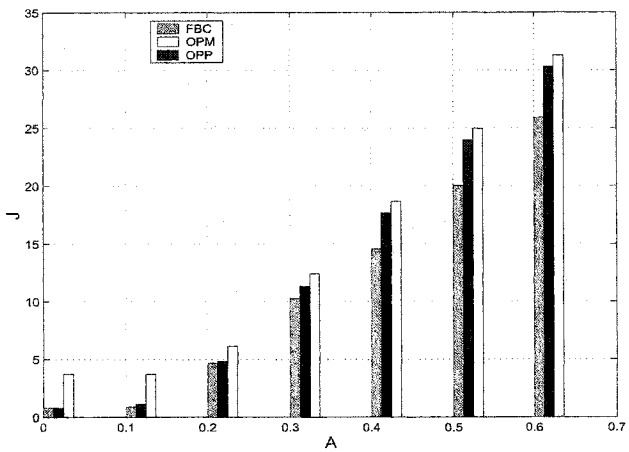

Figure 8.4: Cost vs. A

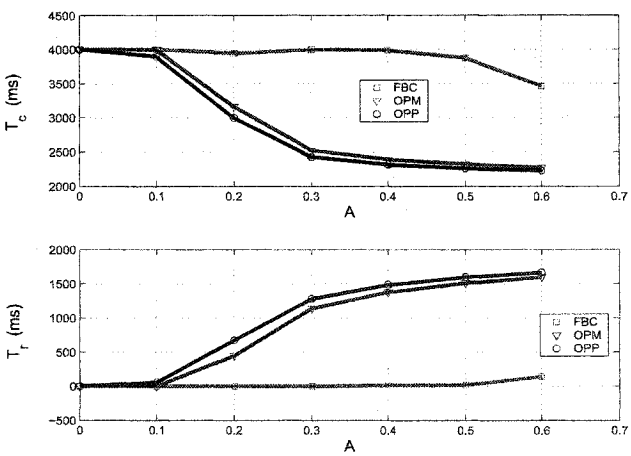

Figure 8.5: Congestion vs. A

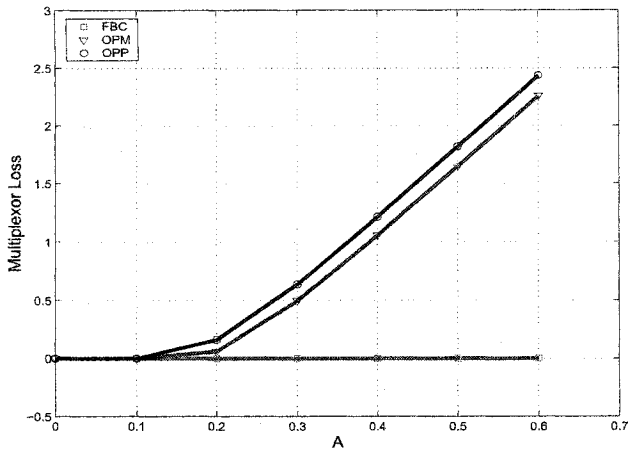

Figure 8.6: Multiplexor Loss vs. A

cial for system performance. Clearly, larger variation increases the time spent below this rate thereby degrading the performance.

The feedback policy, FBC, dynamically adjusts the token generating rate based on available system resources. Figure 8.5 shows the performance in regards to congestion. With the increase of the parameter A both the open loop policies suffer congestion, decreasing the first time to congestion and increasing the residence time in the state of congestion. In contrast, the feedback policy, FBC, maintains good performance over a large range of variation of A unless it is too large. These results are due to the feedback policy keeping the multiplexor losses at a low level by providing control actions according to available resources while the open loop policies are blind. This is clearly evident in Figure 8.6.

Dependence of Performance on Frequency of Channel Capacity Variation

In Figure 8.7, the costs are plotted against the frequency (f) of the channel variation. Again, the feedback control performs better than open loop controls. It is observed that

the cost declines as f increases.

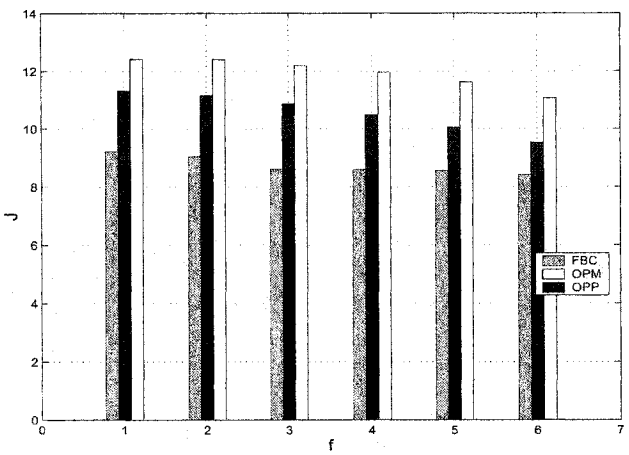

Figure 8.7: Cost vs. Frequency f

8.3 Simple Feedback Control Parameters Chosen by Experiments

Although FBC usually results in a low cost, to evaluate a set of optimized parameters with the simulated annealing algorithm is a time consuming process. In our experiments, we found the cost was not sensitive to k_1, k_2 and λ in the optimal set of parameters, small values are suggested to take in this instance. k_3 is almost half of the channel capacity C. β values are around 20 in many sets of optimal parameters.

8.4 Neural Network Approximation Results

We apply the simulated annealing algorithm to search the space of the weights $W \in R^{n(m+\ell)}$ for the minimal cost and its corresponding set of weight W^o. For the experiment,

we choose the traffic with $r = 0.10,\ 0.14$ (r is a constant that determines the variance of traffic rate in equation 6.6) respectively.

We use the neural networks with the obtained optimal weight W^o (Appendix A) to allocate resource to corresponding traffic. For stochastic traffic we are interested in expected values. As we discussed in section (7.2), the traffic with larger variance requires more sample paths to reach a converged cost value. Based on Figure 7.1, we use 600 simple paths for the $r = 0.14$ traffic. As shown in Figure 8.8 and 8.9, compared with FBC and open loop control strategies, the control strategy applying neural networks have the lowest costs (NN). As we expected, using the same control strategy, the traffic rate with a larger variance has a higher cost. It is important to note that the optimal weight depends on traffic variance, thus, one should evaluate this again whenever the variance parameter changes. Here we have used full information a, ρ, q as the neural network inputs. This is costly.

In case we used only partial information, such as the state information ρ, q as neural network inputs, the optimal cost (NNS), as shown in Figure 8.8 and 8.9, is inferior to that corresponding to full information NN but superior to those corresponding to FBC, OPP and OPM. This result is practically important since it uses only state feedback and does not require any direct measurement of the incoming traffic.

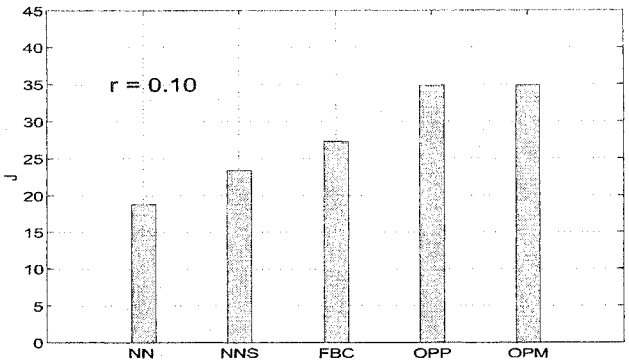

Figure 8.8: Cost vs. Control Policies ($r = 0.1$)

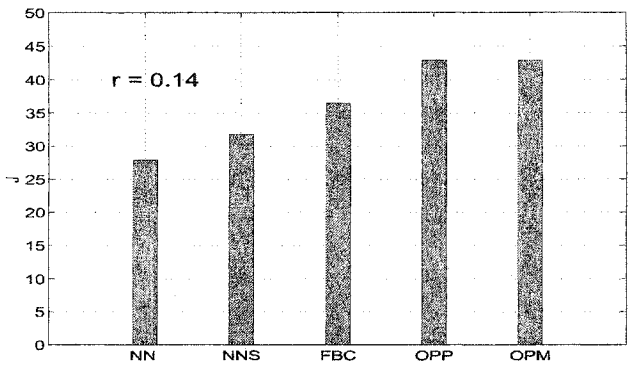

Figure 8.9: Cost vs. Control Policies ($r = 0.14$)

Chapter 9

Conclusion and Future Work

9.1 Conclusion

Firstly, we have developed stochastic traffic simulation models based on standard Brownian motion and fractional Brownian motion. The models can capture the network traffic characteristics of self-similarity, and the model based on fractional Brownian motion can also capture long range dependence (LRD) as observed in real networks, and they are easily implementable. The models are then coupled with the system describing the dynamics of the token bucket access control mechanism and the multiplexor. An objective functional reflecting the loss at TB and multiplexor, including the service delay, is used to evaluate the performance of the system.

Secondly, in order to improve the system performance, we proposed a simple feedback control law based on some simple functions regarding the available space and change rate of the available space of the multiplexor. To prevent the system from trapping in local minima, we used the simulated annealing algorithm to optimize the parameters of the control law. The numerical results demonstrate that the cost is decreased, and the probability of congestion occurring is greatly reduced.

Thirdly, we also proposed a feedback control based on a neural network with incoming traffic and the system state as the input and control as the output, we made use of the simulated annealing to search a set of weights that lead to the lowest cost. Although this set of weights may not be the optimal set, the outcomes show that the cost can be further reduced.

Finally, to avoid those negative values that make no sense in practice, we apply the adaptive time interval step method. With this method, those negative values are completely eliminated while not causing a significant increment of computing time. The method may be useful for other applications in solving similar problems.

9.2 Future Work

Future work which could be of interest are: (1) to develop other simple feedback control strategies that may provide better performance compared to the one suggested; (2) to develop a predictive control law that can estimate future state (as long as there is a feedback delay) and provide control actions reducing performance degradation due to delay, which is not considered in the thesis; (3) to improve the feedback control based on neural network. By applying neural network and simulated annealing, the system displays great performances. However, it takes too much computing time to find an optimal set of neural network weights, and the found set is not robust enough, further research may lead to even better results.

Bibliography

[1] N.U. Ahmed, K.L. Teo, *Dynamic Models For Computer Communication Networks And Their Mathematical Analysis*, Dynamic of Continuous, Discrete and Impulsive Systems Series B: Application & Algorithms 9 (2002) 507-524

[2] N.U. Ahmed, Qun Wang, L. Orozco Barbosa, *A Systems Approach to Modeling The Token Bucket Algorithm In Computer Networks*, Mathematical Problems in Engineering: Theory, Methods and Applications, 2002, 8(3), 265-279.

[3] Shenker, S. and Wroclawski, A. and Callon, R. *Multi Protocol Label Swithing Architecture*, RFC 3031, 2001

[4] Heinanen, J. and Guerin, R. *A Single Rate Three Color Marker*, RFC 2697, 1999

[5] Heinanen, J. and Guerin, R. *A Two Rate Three Color Marker*, RFC 2698, 1999

[6] Aboul-Magd, O. and Jamoussi, B. *InformationalA Framework for Service Definition and Inter Working Using CR-LDP*, Internet Draft, draft-aboulmagd-srvc-def-crldp-00.txt, October 1999

[7] Butto, M., Cavallero, E., Tonietti, A. *Effectiveness of the Leaky Bucket Policing Mechanism in ATM Networks*, IEEE Joural on Selected Area in Communications, 9(3), 1991

[8] Natarajan Gautam, *Buffered and Unbuffered Leaky Bucket Policing: Guaranteeing QoS, Design and Admission Control*, Telecommunication Systems 21:1, 3563, 2002

[9] L. Gun, V.G. Kulkarni and A. Narayanan, *Bandwidth allocation and access control in high-speed networks*, Annals of Operations Research 49 (1994) 161183.

[10] K. Sohraby and M. Sidi, *On the performance of bursty and modulated sources subject to leaky bucket rate-based access control schemes*, IEEE Transactions on Communications 42(24) (1994).

[11] Cisco, *Policing and Shaping Overview*, http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/qos_c/qcpart4/qcpolts.htm

[12] Matt Welsh and David Culler, *Overload Management as a Fundamental Service Design Primitive*, The Tenth ACM SIGOPS European Workshop, Saint-Emilion, France, September, 2002

[13] N.U. Ahmed, Bo Li and L.Orozco Barbosa, *Optimization of Computer Network Traffic Controllers using a Dynamic Programming/Genetic Algorithm Approach.* (submitted)

[14] N.U. Ahmed, Hong Yan and L.Orozco Barbosa, *Performance Evaluation of Access Control Mechanism Corresponding to Stochastic Inputs*, Dynamic of Continuous, Discrete and Impulsive Systems (submitted).

[15] H. Zhang, O. Yang and H. Mouftah, *A Hop-by-hop Flow Controller for a Virtucal Path*, Computer Networks , Vol. 32 (2000), pp.99 - 119

[16] Vishal Misra, Wei-Bo Gong, and Don Towsley *Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application*, SIGCOMM, Stockholm, Sweden, 2000

[17] V. Guffens, G. Bastin, H. Mounier, *Using token leaky bucket with feedback control for guaranteed boundedness of buffer queue*, preliminary work for submission to ECC 2003

[18] Kevin Thompson, Gregory J. Miller, and Rick Wilder, *Wide-Area Internet Traffic Patterns and Characteristics*, IEEE Network, November/December 1997

[19] Mine Caglar, K. R. Krishnan, and Iraj Saniee, *Estimation of Traffic Parameters in High-Speed Data Networks*, Sixteenth International Teletraffic Congress Edinburgh, UK, 7-11 June,1999

[20] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, *On the Self-Similar Nature of Ethernet Traffic (Extended Version)*, IEEE/ACM Transactions on Networking, 2(1), pp. 1-15, February 1994

[21] V. Paxson and S. Floyd, *Wide-area traffic: the failure of Poisson modeling*, IEEE/ACM Transactions on Networking 3,pp.226-244, 1994.

[22] M. Crovella and A. Bestavros, *Self-similarity in World Wide Web traffic: evidence and possible causes*, IEEE/ACM Transactions on Networking 5,pp. 835-846, 1996

[23] Jim Roberts, *Traffic theory and Internet traffic engineering*, IEEE Communications, January 2001

[24] W. S. Cleveland, Don X. Sun, *Internet Traffic Data*, Technical report, Bell Labs

[25] H.-D. J. Jeong, D. McNickle and K. Pawlikowski, *A Comparative Study of Three Self-similar Teletraffic Generators*, Proc. European Simulation Multiconference ESM'99, Warsaw, International Society for Computer Simulation, June 1999, pp. 356-362

[26] Kai-Lung Hua, *On Generator of Network Arrivals with Self-Similar Nature*, on-line `http://shannon.cm.nctu.edu.tw/html/thesis/kai-lung-s.pdf`

[27] Harold J. Kushner, *Analysis of Controlled Multiplexing Systems via Numerical Stochastic Control Methods*, IEEE Journal on Selected Areas in Communications, Vol. 13, No. 7 September 1995

[28] Sally Floyd and Vern Paxson, *Difficulties in Simulating the Internet*, IEEE/ACM Transactions on Networking, February 2001

[29] Hae-Duck J. Jeong, Krzysztof Pawlikowski, Donald C. McNickle, *Generation of Self-Similar Time Series for Simulation Studies of Telecommunication Networks*, First Western Pacific and Third Australia-Japan Workshop on Stochastic Models in Engineering, Technology and Management, pages 221-230, Christchurch, New Zealand, 1999

[30] Tatsuya Hagiwara, Hiroki Doi, Hideki Tode, Hirromara Ikeda, *High-Speed Calculation Method of the Hurst Parameter Based on Real Traffic*, 25th Annual IEEE Conference on Local Computer Networks (LCN'00)

[31] Real traffic data, www.ensc.sfu.ca/ ljilja/ENSC894/

[32] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V.Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, *Recommendations on Queue Management and Congestion Avoidance in the Internet*, RFC 2309, Network Working Group, IETF April 1998

[33] N.U. Ahmed, *Linear and Nonlinear Filtering for Scientists and Engineers*, World Scientific, 1998, pp. 10-11

[34] Philippe Carmona, Laure Coutin, *Fractional Brownian motion and the Markov Property*, Electronic Communications in Probability, 3 (1998) 95-107

[35] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, X. Xiao, *Overview and Principles of Internet Traffic Engineering*, RFC 3272, Network Working Group, IETF, May,2002

[36] A. Dubi, *Monte Carlo Applications in Systems Engineering*, John Wiley, Sons Ltd, NY, USA 2000

[37] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M. N., Teller, A.H. and Teller, E., *Equations of State Calculations by Fast Computing Machines*, J. Chem. Phys. 21, 1087- 1092, 1958

[38] Kirkpatrick, S , Gelatt, C.D., Vecchi, M.P. *Optimization by Simulated Annealing*, Science, vol 220, No. 4598, pp671-680, 1983

[39] Stephane Moins, *Implementation of a simulated annealing algorithm for Matlab*, www.ep.liu.se/exjobb/isy/2002/3339/exjobb.pdf

[40] Goffe, Ferrier and Rogers, *Global Optimization of Statistical Functions with Simulated Annealing*, Journal of Econometrics, Vol. 60, No. 1/2, Jan./Feb. 1994, pp. 65-100

[41] Nickhil Jakatdar, Xinhui Niu, Costas J.Spanos, *A Neural Network Approach to Rapid Thin Film Characterization*, Flatness, Roughness and Discrete Defects Characterization for Computer Disks, Wafers and Flat Panel Displays II LASE-98 Precision Manufacturing Technologies Photonics West, SPIE, January 1998

[42] Robert Morris, Dong Lin, *Variance of AggregatedWeb Traffic*, IEEE INFOCOM 2000 Conference

[43] N.U.Ahmed, Cheng Li, *Stochastic Models for Traffic Dynamics and Access Control Mechanism*, Computer Networks (submited)

[44] Ivan Galkin, U. MASS Lowell, *Crash Introduction to Artificial Neural Networks*, http://ulcar.uml.edu/ iag/CS/Intro-to-ANN.html

[45] George L. Nemhauser, *Introduction to Dynamic Programming*, John Wiley and Sons, Inc., New York, pp. 1-14, 1966

[46] Howard Demuth, Mark Beale, *Neural Network Toolbox For Use with MATLAB*, www.mathworks.com

[47] Harold J. Kushner, *Analysis of Controlled Multiplexing Systems via Numberical Stochastic Control Methods*, IEEE Journal on Selected Areas in Communications, Vol. 13, No.7, September 1995

[48] Desmond J. Higham, *An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations*, Siam Review Vol. 43. No. 3 pp.525-546

[49] Encyclopedia Brittanica, *Brownian Movement*, 1968

Appendix A Neural Network Weights

For the first case (r=0.1), the weights and biases are:

$$W^I = \begin{bmatrix} 0.3449 & 0.5006 & 0.8403 & 0.5445 & 0.4614 & 0.8192 & 0.3514 \\ 0.8541 & 0.2998 & 0.8630 & 0.0832 & 0.4599 & 0.0276 & 0.2420 \\ 0.6250 & 0.7098 & 0.3846 & 0.5439 & 0.5611 & 0.0340 & 0.4816 \end{bmatrix}$$

$$W^L = \begin{bmatrix} 0.5143 & 0.6448 & 0.5987 \\ 0.9512 & 0.0685 & 0.3637 \\ 0.2370 & 0.8099 & 0.5780 \end{bmatrix}$$

$$b^1 = \begin{bmatrix} -0.7325 \\ 0.9137 \\ 0.6558 \end{bmatrix}$$

$$b^2 = \begin{bmatrix} 0.7712 \\ 0.8341 \\ -0.4127 \end{bmatrix}$$

For the second case (r=0.14), the weights and biases are:

$$W^I = \begin{bmatrix} 0.3303 & 0.3019 & 0.2448 & 0.9383 & 0.7838 & 0.7475 & 0.8287 \\ 0.3110 & 0.9304 & 0.4570 & 0.2413 & 0.8617 & 0.3146 & 0.4316 \\ 0.0450 & 0.4248 & 0.4284 & 0.8749 & 0.2323 & 0.3104 & 0.2572 \end{bmatrix}$$

$$W^L = \begin{bmatrix} 0.9331 & 0.4463 & 0.3080 \\ 0.6240 & 0.0573 & 0.1107 \\ 0.8680 & 0.5692 & 0.7645 \end{bmatrix}$$

$$b^1 = \begin{bmatrix} -0.4903 \\ 0.9312 \\ -0.1890 \end{bmatrix}$$

$$b^2 = \begin{bmatrix} 0.4693 \\ -0.9065 \\ 0.8851 \end{bmatrix}$$