Network Traffic Modelling and Router Performance Optimization Using Fuzzy Logic and Genetic Algorithms

BY

Abul Kashem Muhammed Ziaur Rahman

THESIS

Submitted to the Faculty of IT
University of Technology Sydney (UTS)
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Computing Sciences

"I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.
I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis."

Zia A. Rahman
Sydney, Australia.

Abstract

Accurate computer network traffic models are required for many network tasks such as network analysis, performance optimization and areas of traffic engineering such as avoiding congestion or guaranteeing a specific quality of service (QoS) to an application. Existing traffic modelling techniques rely on precise mathematical analysis of extensive measured data such as packet arrival time, packet size and server-side or client-side round trip time. With the advent of high speed broadband networks, gathering an acceptable quantity of data needed for the precise representation of traffic is a difficult, time consuming, expensive and in some cases almost an impossible task. A possible alternative is to employ fuzzy logic based models which can represent processes characterized by imprecise data, which is generally easier to gather. The effectiveness of these models has been demonstrated in many industrial applications. This work develops fuzzy logic based traffic models using imprecise data sets that can be obtained realistically. Optimizing the performance of a router requires the optimization of a number of conflicting objectives. A possible approach is to express it as a multi-objective problem. Multi-objective evolutionary algorithms (MOEA) can be used for solving such problems. This research proposes two fuzzy logic based traffic models: fuzzy group model and fuzzy state model. These models together with MOEA are used to propose a simple and fast router buffer management scheme. The developed fuzzy group model includes a parameter which is also useful for measuring the irregular traffic patterns known as burstiness. The experimental results are promising.

Acknowledgments

This thesis is the outcome of years of hard work. During this long journey, I have been supported and motivated by many people. It is a pleasant feeling that now I have the opportunity to express my gratitude for all of them.

Foremost, I would like to express my deep gratitude to my principal supervisor, Professor Jenny Edwards. She has shown me the way of research. She has always been available to advise me. Besides of being an excellent supervisor, she is a great motivator. During the past five years, Jenny was as close as a relative or good friend to me. Without her constant support this thesis would never come to light.

I would like to thank my co-supervisor Dr Paul Kennedy who kept a constant eye on my progress. He helped me immensely to develop my writing and communication skills. He was always my last resort in solving the most difficult problems. He could not even realize how much I have learned from him.

Special thanks to Dr Andrew Simmonds, my other co-supervisor. His immense knowledge on computer networks and cordial but valuable comments made me look inside of the complex area of computer networks.

I feel a profound sense of gratitude for my late father (whom I lost during the middle of my research) and mother who taught me all the good things that really matter in life. The happy memory of my father is still the persistent source of inspiration for my journey in this life. I am grateful to my brother and sisters. I am glad to be one of them.

I am very grateful to my wife Curie, for her love and patience during the last five years. One of the greatest moments that we had in this period was the birth of our only child, Ryan, who provided a new joyful dimension to our life. Thanks Ryan, life would be incomplete without you.

Dedicated to Ryan M Rahman
My only son (and sun)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet is a complex network [105, 133]. In a competitive market place, network designers, planners and managers need to analyze and optimize the performance of networks to guarantee a specific Quality of Service (QoS) to end-users and to avoid network congestion resulting from variations in the rate of flow of user traffic. As Internet user bandwidth heads toward the Gigabit per Second (Gbps) range, network designers need to reexamine: high performance network protocols, routing protocols, switching techniques, congestion control mechanisms, network measurement, analysis and management tools, network topologies. All of these technologies require appropriate traffic modelling.

Existing modelling techniques (discussed in Chapter 2) rely heavily on precise mathematical analysis and extensive measured data [65]. With the advent of high-speed broadband networks, gathering an acceptable quantity of data needed for precise representation of the traffic is difficult, time-consuming and even, in some cases, an impossible task. An alternative might be to use fuzzy logic based models which can represent systems characterized by imprecise data.

In recent years, a variety of new applications such as video streaming and network interactive games have been introduced on the Internet. With these new applications, the characteristics of Internet traffic (such as packet inter-arrival time, the distribution of packet sizes and application protocols) have also changed [123]. Hence new network modelling approaches are needed to deliver efficient and sophisticated network router buffer management schemes that can avoid congestion and still be able to deliver a specific quality of service.

Buffers in routers are valuable resources. For optimum performance, routers may temporarily store or even drop packets. Managing a router buffer requires the optimization of several conflicting objectives. For example, as the packet drop rate decreases, the queuing delay may increase. The optimization of computer network systems involves a number of such problems which are known as multi-objective optimization problems.

In recent years Multi objective evolutionary algorithms (MOEAs) have received much interest for addressing these problems. These algorithms aim to encapsulate the possibly complex interactions of the various objectives of problems in such a way that the true dynamics and the full potential of the system being optimized may be explored.

The main objectives of this study are to:

1. develop appropriate traffic modelling and performance analysis schemes for high-speed computer networks. The approaches taken should allow reasonably accurate models of the traffic to be developed using data sets that can be obtained realistically. Such models should be simple to implement but still useful in traffic analysis (such as queuing performance analysis and burstiness analysis) and performance improvement.
2. explore the suitability of fuzzy logic for building such models.
3. develop a mathematical quantity to measure the burstiness of a traffic stream.
4. explore the application of MOEAs for optimizing aspects of network performance such as router buffer queue management.
5. explore how the developed traffic models can be applied to examine other network problems such as traffic analysis.

This thesis has eight chapters. **Chapter 2** describes existing traffic modelling schemes. Existing traffic models are classified into three generations. The first generation models are Poisson based models suitable for modelling traditional tele-traffic. Second generation models aim to model data-traffic such as the early Internet traffic. Second generation models focus on capturing the self-similarity nature of network traffic. Recently, several researchers observed non-stationarity in modern Internet traffic making previous models inadequate (all previous models assume network traffic to be stationary). The models that focus on the non-stationary characteristics of traffic are termed third generation models. Chapter 2 also discusses the need for a new approach to modelling network traffic. It is argued that fuzzy logic based models are appropriate to model modern day heterogeneous traffic.

Chapter 3 discusses some network measurement tools, various network performance parameters and network analysis and how they can be used in the complex problem of dimensioning a router buffer. This chapter also describes Active Queue Management schemes such as Random Early Detection (RED) and some of its variations for managing the router buffer queue and the network congestion caused by the in-built TCP flow control mechanism and variation in the rates of flows. The router queue management is formulated as a Multi-Objective Optimization (MO) problem and the design principles for the queue management scheme is discussed.

Chapter 4 reviews soft computing and in particular: fuzzy logic and evolutionary computing. Soft Computing (SC) tries to resemble human reasoning. Unlike conventional computing (hard computing) which strives for exactness and relies on the mathematical capabilities of a computer, SC is tolerant of uncertainty, approximation and partial truth. The first part of this chapter reviews fuzzy logic and fuzzy logic systems. The second part of this chapter discusses the available solution methodologies (classical approach and evolutionary approach) for a MO problem. The idea of Pareto optimality and how it may be used to find an acceptable solution of a MO problem is explored in this chapter.

Chapter 5 contains our proposed fuzzy logic based traffic models. Two such models are proposed. The first named *fuzzy state model* models the state of a traffic stream. This model can be used in a router to schedule its resources and to set up its control parameters (such as packet marking/dropping rate). The other model named *fuzzy group model* is developed to assist performance analysis. This model introduces the *R Parameter* which can have multiple applications is used in the implementation of these models.

Chapter 6 discusses the analysis of 84 publicly available network traffic traces using our fuzzy logic based traffic models. The value of the R parameter and H

parameter (Hurst Parameter) of these 84 traces are computed and their relationship with queuing performance and burstiness is discussed. Queuing performance is analyzed with the help of a trace-driven queuing simulator and the burstiness is analyzed with the help of energy plots. It is shown that the R parameter can be a valuable tool for exploring traffic characteristics.

Chapter 7 proposes Fuzzy Adaptive Fair Random Early Detection (FAFRED), an active queue management scheme. FAFRED is developed based on our proposed traffic models. Various router performance indices, such as packet drop rate, average queue length and fairness indices are computed for FAFRED and the results are compared with other active queue management schemes. The comparison results are promising, indicating that the performance of a router can be improved using FAFRED.

Chapter 8 contains concluding remarks. The results and limitations of this research work are discussed. Several recommendations for future research are made.

Technical terms are defined in the text, where necessary and appear in the glossary (Appendix A). List of abbreviations and symbols used are given in Appendix B and F respectively.

Chapter 2

Historical View of Network Traffic Models

In this chapter models for characterizing Internet traffic are discussed. Section 2.1 briefly introduces the Internet, network traffic and traffic models. Most existing traffic models are statistical and mathematical models. Section 2.2 to Section 2.8 reviews existing (statistical and mathematical) traffic models from a historical perspective. Section 2.9 summarizes the weaknesses of statistical models and describes the motivation on developing traffic models using a different approach. One aim of this research is to develop a relatively simple but accurate traffic model that can capture complex characteristics of network traffic. One such characteristic is the burstiness. Section 2.10 discusses existing techniques to measure the burstiness of a traffic stream.

2.1 Internet Traffic

The Internet is built upon the TCP/IP protocol suite [30]. In this context, a protocol is an agreed set of rules for data interchange. TCP/IP is a set of protocols particularly designed for inter-networking. An internetwork is a series of more than one network interconnected by routers (or other devices) that functions as a single logical network.

On the Internet, millions of communications similar to the above scenario take place every second. These communications make up the flow of a very large number of packets over the physical media. This flow is known as Internet traffic. Internet traffic can be classified into two types: user traffic and aggregate traffic.

User traffic is the traffic produced by a user application. Aggregate traffic is the aggregation of more than one stream of user traffic. That is, the traffic is aggregate traffic when several user traffic streams are aggregated together and transmitted over a common shared media. Figure 2.1 shows a block diagram of user and aggregate traffic. User traffic is characterized by a *network flow*. A network flow or simply a *flow* is a stream of packets of the same application transmitted from the source to the destination in a single session. Clear understanding of the characteristics of both aggregate and user traffic is critical to the design and performance of a network. The characteristics of Internet traffic are described mathematically by traffic models which are reviewed next.

2.2 Application of Traffic Models

A traffic model is a tool to analyze the ways in which traffic packets move from host to host on the Internet. In recent years, a variety of new applications such as video streaming and network interactive games have been introduced on the Internet. With these new applications, the characteristics of Internet traffic (such as packet inter-arrival times, packet sizes and application protocols) have also changed.

Owing to the growth of the Internet, a router on the Internet might need to drop a number of packets due to overflow of the router buffer. The situation, when incoming traffic is so heavy that router buffer overflow occurs, is called *congestion*. A network can become congested due to other reasons such as two or more packets arriving at a node and wanting to exit from the same output interface at the same time. The best way to tackle congestion is congestion avoidance [61]. An alternative can be adopting a more resilient network architecture with built-in optimum buffer capabilities. Any successful congestion control or avoidance

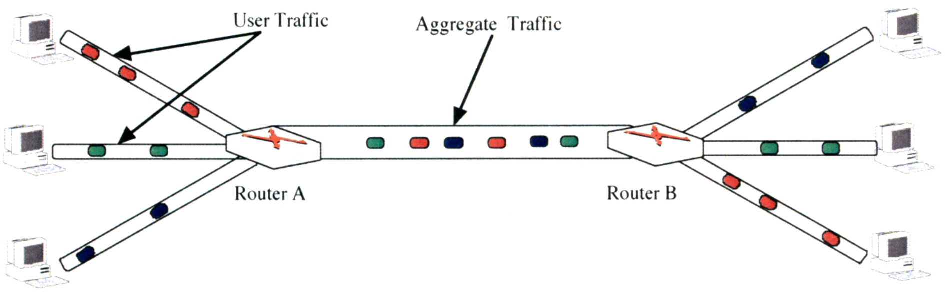


Figure 2.1: User Traffic and Aggregate Traffic

algorithm requires an appropriate traffic model.

Modelling Internet traffic is a complex task as traffic increasingly contains not only text messages or numerical data but also audio and video broadcasting. Modelling video traffic is more complex simply because the activity level is high and there is a higher variability in the packet size and the arrival distribution of packets [12]. In this integrated multi-service, multimedia networking environment, one of the most important aspects of the design and management of networks is knowledge of the characteristics of network traffic. A traffic model should be able to capture these complex traffic characteristics so that it can be useful for many areas of computer networking, such as congestion notification and avoidance, traffic control and management, dynamic routing and control, network resource scheduling, network performance analysis and Quality of Service (QoS) based routing and billing. Some of these applications are discussed in Chapter 3.

Traffic control, management and network resource scheduling are achieved by means of queue management schemes. Traffic control and queue management systems are a set of mechanisms by which packets are received, queued (if required) and forwarded to the appropriate interface of the router. In a queue management system, generally, if the network becomes too congested, packets may be dropped instead of being queued. The packet drop rate is dynamic and depends on the incoming traffic pattern and the network congestion state. The design of these systems depends on the traffic model. For example, a traffic control system that assumes traffic follows the Poisson distribution will require a smaller buffer size and a smaller packet drop rate than a model that assumes the traffic to be non-Poisson [2]. This will be discussed further in the later sections of this chapter.

One single traffic model might not be appropriate for all areas where an accurate traffic model is required. The aim of our research is to develop one or more traffic model(s) that can be used to analyze the complex characteristics of Internet traffic and to apply the model(s) to develop a queue management scheme with excellent performance.

2.3 Mathematical Representation of Traffic Streams

A simple traffic stream is a sequence of arrivals of discrete entities such as packets or cells. Mathematically a traffic stream can be described in one of the following three ways [65]:

1. as a *point process*
2. as a *counting process*
3. as an *inter-arrival time process*

A traffic can be mathematically described as a point process consisting of a sequence of packet arrival times $T_1, T_2, \cdots, T_n, \cdots$ measured from the origin. It is a convention to assume the origin T_0 to be 0.

In a counting process representation, a traffic stream is expressed as a counting process $\{N(t)\}_{t=0}^{\infty}$. This process is a continuous-time, non-negative integer valued stochastic process, where $N(t)$ is the number of traffic (such as packet or byte) arrivals in the interval $(0, t)$.

In an inter-arrival time process representation, a traffic stream is expressed as a inter-arrival time process, which is a real-valued random sequence $\{A_n\}_{n=1}^{\infty}$, where $A_n = T_n - T_{n-1}$ is the length of the time interval between the *n-th* and the previous arrivals.

An alternative to the above representations is a binning approach [74]. In this approach the duration of the traffic stream is divided into a finite number of non-overlapping time bins of equal length. Each bin is associated with a start time and an end time. The start time of a bin is equal to the end time of the previous bin and the end time of a bin is equal to the start time of the following bin. A traffic stream is then represented as a real-valued non-negative sequence $\{N(i)\}_{i=1}^{\infty}$, where $N(i)$ represents the number of traffic (such as packet or byte) arrivals in the *ith* time bin.

All the above representations express a traffic stream as a time series. Statistical models have long been used for modelling time-series. Consequently most existing traffic models are statistical models. Generally in a statistical model, the model for the time-series is developed first and then the model parameter(s) are adjusted to fit the measured data. But if the nature of the time-series is complex and if the statistical properties of the time-series are changing over time, it may not always be possible to fit the model parameters accurately. An alternate approach could be to build the models directly from the measured data.

Internet traffic models from a historical perspective are reviewed next. In this work, based on the time of development, traffic models are classified into three generations. Traffic models in each generation take different approaches and make different assumptions. The review discusses the limitations of existing models and the overall statistical approaches they use. In section 2.9, motivations for developing traffic models that use a measured data approach and do not make any statistical assumptions are discussed.

2.4 First Generation Models - Non-Self-Similar Models

Traffic models developed before the 1990s focused mainly on simplicity of analysis. They assumed non-self-similarity where a working definition of self-similarity is that a self-similar object is exactly or approximately similar to a part of itself. It looks or behaves the same when viewed at different scales on a dimension such as space or time. The models developed before the discovery of the existence of self-similarity in network traffic are classified as **First Generation Models.**

Informally, burstiness means that the number of packets arriving in a short time interval is much higher than the average. First generation models assume that aggregated traffic would smooth out these traffic bursts as the number of traffic sources increased. The earliest such models were based on Poisson distributions and ignored bursts completely. Later, burstiness was observed on network traffic and attempts were made to incorporate burstiness on top of Poisson distributions. The pure Poisson models are reviewed first.

2.4.1 Poisson Models of Internet traffic

The Poisson distribution expresses the probability of a number of events occurring in a fixed period of time, provided the events occur with a constant average rate and the occurrence of these events is independent of each other. If an event follows a Poisson distribution, the probability that there are exactly k occurrences of the event during a time interval is given by

$$f(k,\lambda) = \frac{e^{-\lambda}\lambda^k}{k!} \tag{2.1}$$

where k is a non-negative integer and λ is the average occurrence rate of the event during the interval [116]. A Poisson process is a stochastic process where the events during a time interval follow a Poisson distribution.

Poisson models are the oldest and the simplest traffic models [49]. This model has only one parameter, the mean arrival rate λ. Packet arrivals during an interval t follow a Poisson distribution with mean $t\lambda$ and packet inter-arrival times follow an exponential distribution [116]. The probability density function of an exponential distribution has the form

$$f(x,\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.2}$$

A pure Poisson model can model only smooth traffic. A variation of pure Poisson models, known as a Compound Poisson Model tried to add burstiness to pure Poisson modelling [67]. In this model, packets arrive in batches. The inter-batch arrival times are exponential and the batch size is geometric which means there can be an infinite range of batch sizes. This model requires two parameters, the batch arrival rate, λ and the batch parameter, ρ with $0 < \rho \leq 1$. In this model,

$$\begin{aligned} &\text{mean number of packets in a batch} &&= \rho \\ &\text{mean inter batch arrival time} &&= \tfrac{1}{\lambda} \\ &\text{mean packet arrivals over time period } t &&= \tfrac{t\lambda}{\rho} \end{aligned}$$

2.4.2 Markovian Models

Ordinary Poisson models exhibit too little burstiness to match real traffic. In order to incorporate burstiness, researchers began to consider the superposition of multiple types of traffic sources, each with a different packet generation rate. The result was the development of Markovian models [58].

A process is said to have the *Markov property* [44], if the future state of the process is conditionally independent of the past states. A *Markov chain* [44] is a stochastic process with the Markov property. At each point in time, the process may have changed states from the previous state, or the system may have stayed in the same state. If the Markov chain is restricted to have two states, it is called a two-state Markov chain.

In a Markovian model, packet arrival is a Poisson process where the instantaneous arrival rate is given by the state of a Markov chain, instead of being constant (as in an ordinary Poisson model). This type of arrival process is called a *Markov-modulated Poisson process* (MMPP). In the case of a two-state Markov chain, the MMPP can be thought of as a switched Poisson process with two different arrival rates λ_1 and λ_2 and sojourn time (time spent in a state) α_1 and α_2. When the process is in the first state, packets arrive with rate λ_1 and if the process is in the other state, packets arrive at a rate λ_2. That is, in a two-state Markovian model, the traffic arrival process is an MMPP process defined by four parameters: λ_1, λ_2, α_1 and α_2. The mathematics of how to derive the parameters of a given traffic trace may be found in [58].

By allowing two states, the MMPP model incorporates limited burstiness on top of a Poisson model. This burstiness is still insufficient to match real traffic [1, 38]. The MMPP model can be extended by allowing more than two states and the resulting process would be burstier than the simple two-state case. However this would increase the mathematical complexity of computation and was not studied by the authors, Heffers and Lucantoni in [58].

The next section discusses "self-similarity", a characteristic of network traffic that was unnoticed in the first generation models, yet seems to have serious implications for designing and planning of high-speed networks. At first self-similarity is discussed and then how this discovery makes Poisson based first generation models obsolete is represented.

2.5 The Discovery of Self-Similarity

Self-similarity was first observed by Mandelbrot [89] in a wide range of physical (and mathematical) systems. He illustrated that it is an inherent property in many irregular and bursty systems. Self-similarity is a property of a particular type of stationary process. In this section stationarity, self-similarity and some of the characteristics and consequences of a self-similar process are discussed.

2.5.1 Stationarity

Stationarity can be described in terms of probability theory [43], which is a branch of mathematics that deals with the phenomena of randomness.

Let $X(t_1), \cdots, X(t_i)$ be a time series. Sometimes $X(t_i)$ is represented by X_i and the time series $X(t_1), \cdots, X(t_i)$ is represented by $X_1, \cdots, X_i$ or simply by $\{X_i\}_{i>0}$. The mean, denoted by μ and the variance, denoted by σ^2 of a random process $\{X_i\}_{i>0}$ are defined by

$$\mu = E[X_i] \tag{2.3}$$

$$\sigma^2 = E\left[(X_i - \mu)^2\right] \tag{2.4}$$

where $E[X_i]$ denotes the expected value of the random variable X_i.

The lag k autocorrelation function, $r(k)$ of the process $\{X_i\}_{i>0}$ is defined as

$$r(k) = \frac{E\left[(X_i - \mu)(X_{i+k} - \mu)\right]}{\sigma^2} \tag{2.5}$$

A time series $X(t_1), \cdots, X(t_i)$ is said to be stationary if the joint distribution of $X(t_1), \cdots, X(t_i)$ is the same as the joint distribution of $X(t_1 + t_0), \cdots, X(t_i + t_0)$ for all $t_1, \cdots, t_i$, t_0. In other words, all the statistics, such as mean, variance and autocorrelation of $X(t_1), \cdots, X(t_i)$ remain unchanged after any time shift.

Another weaker form of stationarity, known as *weak-sense stationarity* or *wide-sense stationarity* (WSS) or *covariance stationarity* [116] requires that the first two moments (mean, variance) and also autocorrelation do not vary with respect to time.

Originally, it was believed for a long time that Internet traffic is a *stationary* time series [32, 125]. The first consequence of assuming the stationarity (or weak-sense stationarity) of Internet traffic is that the overall mean of any statistical property (such as byte counts or packet counts) does not change with respect to time. The same holds for higher moments (at least up to second order), such as variance and covariance. Since these moments remain constant, the stationarity (or weak-sense stationarity) property of Internet traffic indicates stable behavior. It also indicates that Internet traffic is not biased or influenced by the time of day or day of the week.

2.5.2 Self-Similarity

Self-similarity is described with the help of the autocorrelation function (equation (2.5)). Let $\{X_i\}_{i>0}$ be a weak-sense stationary (WSS) process with mean μ and variance σ^2. Let $\{X_i\}_{i>0}$ represent a traffic stream, where X_i is a measurement that represents the number of bytes that have arrived during the *ith* time interval. Each time interval can be thought as a time-bin of size equal to the interval.

Using the process $\{X_i\}_{i>0}$, let us define a new process $\{X_i^{(m)}\}_{i>0}$, obtained by averaging the X_is over non-overlapping blocks of size m i.e.,

$$\begin{array}{lcl} X_1^{(m)} & = & \frac{1}{m}(X_1 + \cdots + X_m) \\ X_2^{(m)} & = & \frac{1}{m}(X_{m+1} + \cdots + X_{2m}) \\ \cdots & \cdots & \cdots \\ X_i^{(m)} & = & \frac{1}{m}\left(X_{(i-1)m+1} + \cdots + X_{im}\right) \end{array} \tag{2.6}$$

The formulation of the process $\{X_i^{(m)}\}_{i>0}$ from $\{X_i\}_{i>0}$ is known as the aggregation and m is the aggregation level [37]. Equation (2.6) shows that the construction of the aggregated process $\{X_i^{(m)}\}_{i>0}$ is equivalent to widening the bin size of $\{X_i\}_{i>0}$ by m times and decreasing the number of bins by m times. If the X_is are random, for a large value of m, each $X_i^{(m)}$ will have very similar values and will tend towards μ. In that case, the autocorrelation function of equation (2.5) will be close to zero. This property is known as *short range dependency* (SRD) [37]. Mathematically, a process is SRD, if for all k,

$$r(k) \rightarrow 0 \qquad \text{as } m \rightarrow \infty \tag{2.7}$$

Equation (2.7) shows that an SRD process has a finite autocorrelation function for all values of k. In other words, it is said that SRD processes have a *summable* autocorrelation function. That is, the summation of autocorrelation functions for all values of k is finite. On the other hand, the autocorrelation function never vanishes for a *Long Range Dependent* (LRD) process [37]. Mathematically, the process $\{X_i\}_{i>0}$ is said to be LRD, if for sufficiently large k ($k \rightarrow \infty$), its autocorrelation function has the form

$$r(k) \sim k^{-(2-2H)} \tag{2.8}$$

with $0.5 < H < 1$. H is called the *Hurst parameter* or simply the H parameter [37]. For $0 < H \leq 0.5$, the process is short range dependent.

The process $\{X_i\}_{i>0}$ is called *exactly second-order self-similar,* if, for sufficiently large k ($k \rightarrow \infty$), the autocorrelation function $r^{(m)}(k)$ of the aggregated process $\{X_i^{(m)}\}_{i>0}$ holds the following condition

$$r^{(m)}(k) \sim k^{-(2-2H)} \text{ for all} \, m \tag{2.9}$$

The process $\{X_i\}_{i>0}$ is called *asymptotically second-order self-similar* or simply *self-similar* if for sufficiently large k ($k \rightarrow \infty$), the following condition holds

$$r^{(m)}(k) \sim k^{-(2-2H)} \quad \text{as } m \rightarrow \infty \tag{2.10}$$

Equations (2.8), (2.9) and (2.10) show that the Hurst parameter completely characterizes LRD or self-similarity. The value of H indicates the degree of self-similarity.

Equation (2.10) shows that, for an asymptotically second-order self-similar process, the autocorrelation decays so slowly with aggregation that no level of aggregation can eliminate the autocorrelation within the process. Subsequent values at large scales ($m \rightarrow \infty$) retain some dependency on prior values even as the scale continues to increase. Because of this property, a self-similar process is called a *long memory* process.

Equation (2.10) shows that the self-similar process has a non-summable autocorrelation function. This implies that, for self-similar traffic, the traffic counts do not smooth out by widening the bin size. That is, burstiness at all time scales is an inherent property of self-similarity.

2.5.2.1 Hurst parameter estimation

Self-similarity is a phenomenon that is observed at large time scales. To observe self-similarity, a huge amount of data is required so that it can be aggregated to large scales (equation (2.6)). Thus, estimating the H parameter for a real traffic trace requires a measurement of huge quantities of data and computation. There are several methods of estimating the Hurst parameter of a self-similar process such as a variance-time plot [32] or the Abry and Veitch method [3]. These are described below.

Variance-Time Plot. Equation (2.10) can be used to estimate the H parameter. The product of the variance and autocorrelation of a process is called the autocovariance of that process. Therefore, denoting the lag by k and the lag k autocovariance by $\gamma(k)$ equation (2.10) can be rewritten as

$$\frac{\gamma^{(m)}(k)}{(\sigma^{(m)})^2} \sim k^{-(2-2H)} \quad \text{as } m \to \infty \tag{2.11}$$

Equation (2.11) shows that the plot of $(\sigma^{(m)})^2$ as a function of m in a log-log scale should be a straight line with slope $2H - 2$. The value of H can be estimated from the slope of the plot.

Abry and Veitch Method. The Abry and Veitch Method [3, 111] computes H by a wavelet decomposition [63] of the time series. In this method a time series consisting of a sequence of packet counts or byte counts in a small time interval (typically in the order of a millisecond) is formed and the *wavelet-spectrum* [63] of this series is generated. The spectrum consists of the wavelet coefficients at each scale. The variance of the coefficients at a scale is plotted against the scale. The value of H is calculated from the slope of the plot [3]. In this work due to their simplicity and accuracy this method is used to estimate H for real traffic traces [25]. The algorithm is outlined in Appendix E and wavelet based energy plots are discussed later (section 2.10.1).

2.5.2.2 Generating Self-Similar Traffic

The two most commonly used self-similar processes are fractional Gaussian noise (fGn) and the fractional autoregressive integrated moving-average (ARIMA) process [96]. Fractional Gaussian noise is exactly self-similar and ARIMA is asymptotically self-similar.

In this work an ARIMA process is generated to compare the queuing performance of our models with real network traces (section 6.3).

ARIMA(0,d,0) is the simplest and the most fundamental of fractional ARIMA processes. ARIMA(0,d,0) is defined as:

$$x_n = \sum_{k=1}^{n} \binom{d}{k} x_{n-k} + \varsigma_n \quad \text{for } n \geq 0 \tag{2.12}$$

where ς is Gaussian White Noise and $\binom{d}{k}$ denotes the binomial coefficient. A Gaussian white noise process is a sequence of independent and identically-distributed (i.i.d.) normal random variables [42]. The parameter d is known as the fractional difference parameter. It is related to the Hurst parameter as

$$d = H - 0.5 \tag{2.13}$$

If d is allowed to have values between -0.5 and 0.5, the process x_n in equation (2.12) is known as a fractional ARIMA (FARIMA) process.

From equation (2.12), for a given value of d, an asymptotically self-similar process can be generated as:

$$\begin{aligned} x_0 &= \varsigma_0 \\ x_1 &= \varsigma_1 + dx_0 \\ x_2 &= \varsigma_2 + dx_1 + \tfrac{1}{2}d(1-d)x_0 \\ &\ldots\ldots \end{aligned} \tag{2.14}$$

In our work equation (2.14) is used to generate self-similar traffic for use in traffic analysis (section 6.3).

2.5.3 Evidence for Self-Similarity in Network Traffic

In the early 1990s, a group of Bellcore researchers found the existence of self-similarity in Ethernet data [124]. In 1994, Leland, Taqqu, Willinger and Wilson in [79] showed the existence of self-similarity on network traces graphically. One of the graphs is given in Figure 2.2 (reprinted with permission of the author). The figure shows that self-similar traffic exhibits high variability at a scale as large as 100 seconds.

Figure 2.2 also shows that at smaller time scales (0.1 second or smaller), both the self-similar and the Poisson traffic have spikes. The spikes mean that the packets

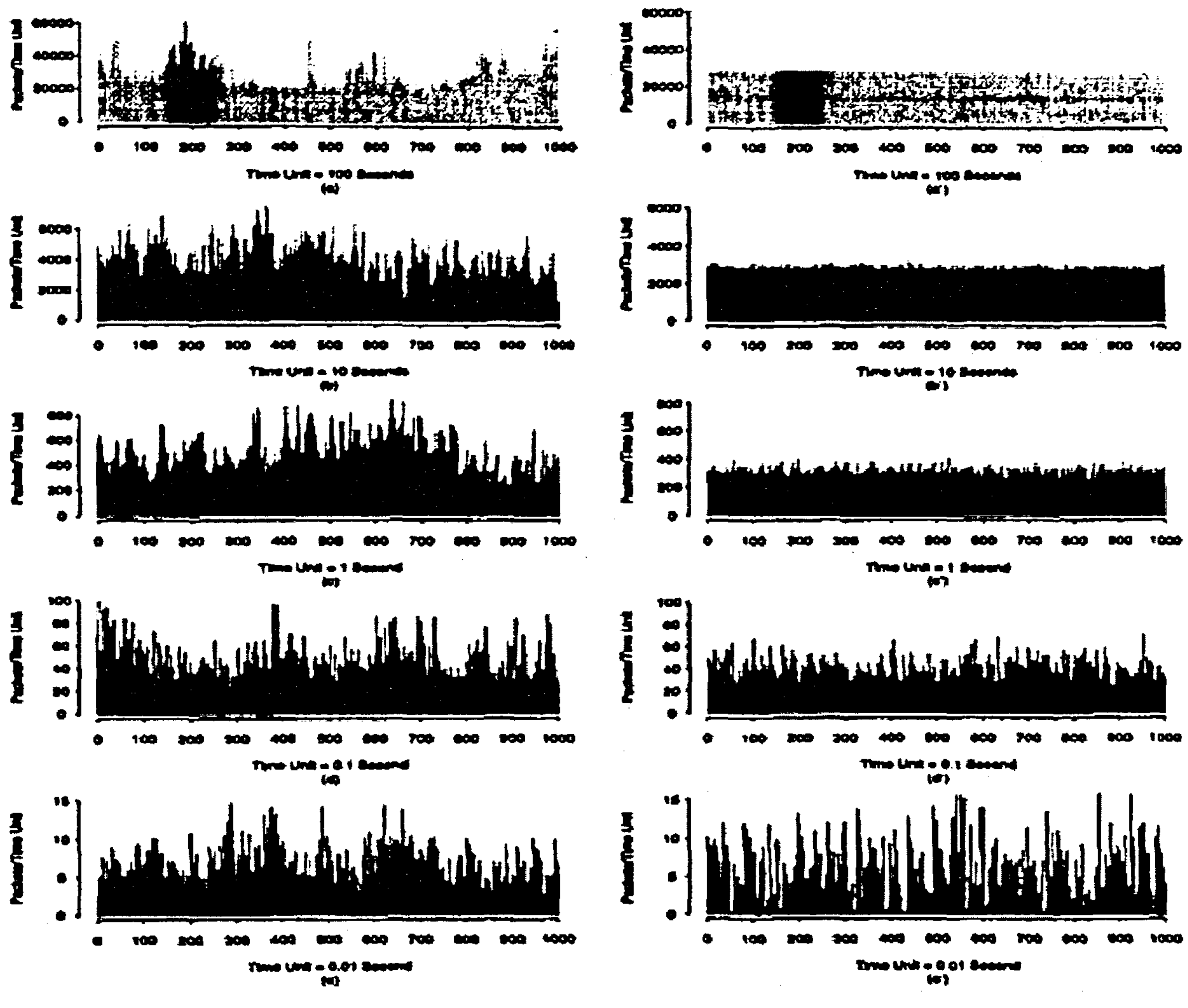


Figure 2.2: Packet count vs time of self-similar network traffic (on left) and Poisson traffic (on right) at various time scales. (Reprinted with permission from authors of [79]). Self-similar traffic exhibits high variability at all time scales.

come in a burst. For Poisson traffic, burstiness disappears and traffic becomes smooth for scales larger than 1 second. However, for self-similar traffic, burstiness does not disappear at scales as high as 100 seconds.

2.5.4 Failure of Poisson Based First Generation Models

All the first generation models are Poisson based and rely on the *Central Limit Theorem*. The Central Limit Theorem states that if more and more random variables with finite variance from a distribution or distributions are aggregated, the aggregate distribution tends to a normal distribution, and the aggregate mean will "smooth" toward a single mean.
Self-similar traffic exhibits LRD, which cannot be removed by aggregation and hence the Central Limit Theorem does not hold. Since all the first generation models rely on Poisson processes, none of them can capture self-similarity. Thus the discovery of self-similarity made all the Poisson based models obsolete.

The theoretical failure of Poisson based models was confirmed in experimental results by Paxson and Floyd [104]. Their study shows the inability of Poisson based models to capture the burstiness of real self-similar traffic at various time scales. Their study also shows that, packet inter-arrivals cannot be modelled by a Poisson distribution. Paxson and Floyd then applied a Pareto distribution [82] to model packet inter-arrival time. Although the Pareto distribution is not self-similar, it can match self-similar traffic for a small period of time [104] up to a time scales of tens of seconds.

Since all the first generation models rely on Poisson processes, they cannot model real network traffic accurately. In a real network, packets may arrive in bursts. As a Poisson process is memoryless, this burstiness cannot be captured by Poisson based models and thus they fail to model the traffic at larger time scales (Figure 2.2). Although the MMPP processes are able to incorporate burstiness, the simple two-state Markov chain is not enough to model real self-similar traffic (section 2.4.2). A number of researchers have shown in practice that Poisson based models, that are termed as first generation are not suitable for modelling self-similar traffic [79, 103, 104].

2.6 Second Generation Models - Self-Similar Models

With the evolution of the Internet, the characteristics of network traffic changed and new models were proposed. In the early to mid 1990s, several researchers confirmed the existence of self-similarity in various local area networks (LAN) and in Internet traffic [32, 54, 79, 103]. In this work, the models proposed in the middle to late 1990s are classified as **Second Generation Models.** These models assume the traffic to be self-similar (and consequently to be covariance stationary). Several popular traffic models such as the chaotic map model and the Brownian motion model were proposed during this period. They are reviewed below.

2.6.1 Chaotic Map Model

The *chaotic behavior* [52] of a process can be characterized by three criteria:

1. It is deterministic, meaning it obeys some simple rule and its future dynamics is well defined by the initial conditions.
2. It is highly sensitive to initial conditions, which means its behavior can be predicted only for short times. Because of this sensitivity, any small change in an initial condition leads to a significant change in future behavior. This property is known as *Sensitive-dependence on Initial Condition* (SIC).
3. It has an underlying pattern.

Chaotic maps [52] are the simplest form of system exhibiting chaotic behavior. A chaotic map is a function with chaotic behavior that maps a state variable to its next state. By iterating the mapping, any number of new states can be generated.

Let us consider a chaotic map $x_{n+1} = f(x_n)$ and two trajectories with nearly identical initial conditions x_0 and $x_0 + \varepsilon$, with $\varepsilon \to 0$. Sensitive-dependence on Initial Condition can be expressed mathematically as

$$|f^N(x_0 + \varepsilon) - f^N(x_0)| = \varepsilon e^{N\xi(x_0)} \tag{2.15}$$

where f^N is the map iterated N times. The parameter ξ that describes the exponential divergence is called the *Liapunov exponent* [40].

A chaotic map model [40] uses chaotic maps to model packet traffic sources. In this model, the state variable x_n evolves over time according to a one-dimensional map:

$$x_{n+1} = \begin{cases} f_1(x_n) & \text{if } 0 < x_n \leq z \\ f_2(x_n) & \text{if } z < x_n < 1 \end{cases} \tag{2.16}$$

where $f_1(.)$ and $f_2(.)$ are chaotic maps satisfying the SIC condition invocation (2.15) and z is the model parameter.

A packet generation process is modelled by assuming that a traffic source is in a passive or active state at time n depending on whether x_n is below or above the threshold z. Each iteration of the map produces a new value of x_n. If x_n is higher than z, then the source is in the ON state and a packet (or a batch of packets where the batch size can be determined by another chaotic map) is generated. When $x_n \leq z$, the source is in OFF state. That is, the packet arrival process is described by y_n as

$$y_n = \begin{cases} 0 \text{ if } 0 < x_n \leq z \\ 1 \text{ if } z < x_n < 1 \end{cases} \tag{2.17}$$

The main challenge in implementing a chaotic map model is to find suitable $f_1(.)$ and $f_2(.)$. Erramilli, Singh and Pruthi in [40] successfully used the Intermittency map to model packet generation. The Intermittency map is defined by

$$x_{n+1} = \begin{cases} \varepsilon + x_n + cx_n^j & \text{if } 0 < x_n \leq z \\ \frac{x_n - z}{1-z} & \text{if } z < x_n < 1 \end{cases} \tag{2.18}$$

where $c = \frac{1-\varepsilon-z}{z^j}$ and j is an integer.

They studied the case of $j = 2$ and showed that the resulting series is self-similar. In simulation, their chaotic map model successfully generated a self-similar process that resembled real TCP traffic with no retransmission. However, they did not attempt to validate these results against real network traffic.

2.6.2 Brownian Motion Model

Norros proposed a traffic model called the storage model [99]. This model is based on *Fractional Brownian Motion* (fBm) which is defined below.

A stochastic process $\{X(t) \,|\, t \in T\}$ is said to be a Gaussian process, if any linear combination of $X(t)$s are normally distributed. That is, for any positive integer $n < \infty$, $a_i \in \mathbb{R}$ and any $t_1, \cdots, t_n \in T$,

$$Y = a_1 X(t_1) + \cdots + a_n X(t_n) \tag{2.19}$$

is normally distributed.

A Gaussian process $\{X(t) \,|\, t \in T\}$, for $0 < H < 1$ satisfying:

1. $X(0) = 0$ and $X(t)$ is continuous
2. for any $t > 0$ and $\alpha > 0$, the increment $X(t + \alpha) - X(t)$ is normally distributed with mean zero and variance α^{2H}

is called Fractional Brownian Motion (fBm), where H is the Hurst parameter. If $H = \frac{1}{2}$, the process is called *Brownian motion.* The derivative of Brownian motion is called *fractional Gaussian Noise* (fGn).

In the storage model, Norros studied a traffic storage system (such as a router) with LRD input. Input traffic is represented by a *Norros process* $V(t)$. Mathematically, a Norros process is a stochastic process and is given by:

$$V(t) = \underset{s \le t}{sup} \left(A(t) - A(s) - C(t-s),\, t \in (-\infty, \infty)\right) \tag{2.20}$$

where *sup* denotes the supremum operator [86] and $A(t)$ is the process

$$A(t) = mt + \sqrt{am} Z(t),\ t \in (-\infty, \infty) \tag{2.21}$$

and $Z(t)$ is a fractional Brownian motion process with self-similarity parameter (Hurst parameter) $H \in \left[\frac{1}{2}, 1\right]$, $m > 0$ is the mean input rate of the Norros process, $a > 0$ is a coefficient of variance and $C > m$ is the service rate of the storage system.

Norros studied the behavior and calculated analytical solutions for traffic originating from the fBm model. In [99], Norros also provided capacity planning and buffer dimensioning solutions for a router to serve self-similar traffic. Informally, these solutions determine the amount of buffer space that is required by a router for optimum performance.

The application of the Norros model to a real network requires estimating the H parameter. The estimation of the H parameter is a difficult task (section 2.5.2.1). The Norros model did not provide any mechanism for estimating the H parameter in a simple manner. This significantly limits the application of the model in real networks.

2.6.3 Discussion of Self-similar Models

Self-similar models are generally attractive as they can capture the burstiness of network traffic. But self-similar models are not always applicable. Erramilli in [39] pointed out the limitations of self-similar models. They demonstrated that self-similar models are appropriate only under the following strict conditions:

- there should be large-scale aggregation of network traffic sources;
- there should not be any significant TCP congestion. Network controls should not significantly impact the flows over the scaling region; and
- the scaling region should be large enough, so that LRD can impact the scaling region. For short samples, LRD might not be reflected.

Another drawback of self-similar models is that the analysis of a queuing system with self-similar traffic is not analytically tractable. Therefore, these models might not be useful in network simulations.

2.7 Evidence of Non-Stationarity

In the early 2000s, a group of researchers from Bell Labs observed pervasive nonstationarity on an uncongested Internet link [15, 16]. They found that, as the traffic load increases, the packet arrival rate tended toward a Poisson arrival rate and the packet size distribution tended toward independence. This was in clear contrast to the self-similarity property previously studied (Section 2.5.2).

Karagiannis, Molle and Faloutsos then studied several backbone (a high-speed line that forms a major pathway within the Internet framework) traces from 2002 and 2003 [71] and found some interesting characteristics, such as:

- at milli-second time scales (up to a few hundred milliseconds), traffic appears to exhibit LRD;
- at sub-second time scales, traffic appears to be Poisson;
- at multi-second time scales, traffic appears smooth, but nonstationary; and
- at a very large scale (such as a few hundreds or thousands of seconds) the global average of a traffic variable (such as byte count) may drift very far away from a local average (indicating nonstationarity).

They found nonstationarity in the Internet traffic in two ways:

1. As the TCP connection rate changes, the parameters of statistical models fitted to the traffic variable, such as the packet (or byte) count distribution change. Example of such parameters are m in the Norros model (equation (2.20)) or j in the chaotic map model (equation (2.18)).
2. As the TCP connection rate changes, the queuing characteristics of the traffic change.

The discovery of nonstationarity makes self-similarity based second generation models inadequate particularly at sub-second or higher scales. This motivates the development of the next generation of models. The models proposed in the early 2000s are classified as **Third Generation Models.**

2.8 Third Generation Models

The observation described above of Poisson patterns at sub-second time scales makes traffic models even more complex. Third generation models try to add a Poisson component to LRD. Such models can be labeled "non-stationary Poisson Models" [71]. The most popular third generation models are the Fractional Sum Difference (FSD) model and its variants which are described below.

2.8.1 The Fractional Sum Difference-Moving Average (FSD-MA) Model

A moving average (MA) process of order k of a time series is the weighted mean of the previous k points. Let, z_u be a first order moving average process, defined by:

$$z_u = \varsigma_u + \varphi\varsigma_{u-1} \tag{2.22}$$

where ς is Gaussian white noise with mean 0 and variance $(1+\varphi^2)^{-1}$. According to the Fractional Sum Difference - Moving Average (1) (FSD-MA(1)) model, traffic is modelled by the process

$$Y_u = \sqrt{1-\theta}x_u + \sqrt{\theta}z_u \tag{2.23}$$

where, x_u is a FARIMA process (equation (2.12)) and θ is the parameter of the FSD-MA(1) model with $0 < \theta < 1$.

Equation (2.23) shows that the traffic process Y_u is a combination of two components. The first component x_u is a FARIMA process. Since a FARIMA process is LRD, the first component of Y_u is the LRD part. The second component of Y_u is z_u, which is a moving average of white noise. Therefore, as $\theta \longrightarrow 0$, $Y_u \approx \sqrt{1-\theta}x_u$. Since x_u is self similar, Y_u is also self-similar with $H = d + 0.5$ (equation (2.13)). As $\theta \longrightarrow 1$, the LRD part equates to zero and Y_u is a first order moving average of white noise. If, $\varphi = 0$ and $\theta = 1$, Y_u is simply white noise and the model is known is known as a Fractional Sum Difference (FSD) model [15]. That is, FSD is a special form of FSD-MA(1).

The FSD-MA(1) and FSD models can capture both the LRD and the Poisson aspects of network traffic. For millisecond and smaller time scales, θ is set to a value close to 0 and the traffic is primarily self-similar traffic. For multi-second and higher time scales, θ is set to a value close to 1.

The FSD-MA(1) model is simple, attractive and promising. Open loop (congestion free) Internet packet traffic [17] and HTTP source traffic [14] are modelled using the FSD-MA model with a great deal of success.

Third generation models are promising, but still in their early stages. The popular FSD model has been applied to model only open loop traffic (section 2.8.1).

2.9 The Need for a New Approach

The observation that, at a multi-second time scale, Internet traffic tends towards Poisson, leads to questions such as: "is the traffic pattern reverting?" or "does LRD exist at higher than multi-second time scales (perhaps a thousand or million second scale)?" or "after how many months or years, will traffic characteristics change again?". The answers are not known. These questions pose challenges to statistical traffic models. The assumptions currently made by statistical models may prove to be wrong with the evolution of networks. For example, first generation models assumed network traffic to be Poisson (section 2.4). Second generation models made the assumption that network traffic is covariance stationary (section 2.6). Later it was found that these assumptions are not quite accurate (section 2.7). These developments provided a motivation for us to develop a traffic model which does not make any such assumptions based on statistical properties.

Generally in the case of statistical models, the models are developed first and then model parameters are adjusted to fit the measured traffic data. An alternate approach can be building the models directly from the measured data. This "measured data approach" has the advantage that unlike statistical models, it does not require any assumptions to be made. Generally this approach is more adaptive than statistical models. The dynamic and complex nature of network traffic makes this approach attractive.

The measured approach has the disadvantage that the measurement of data can add computational complexity. A further complication may arise due to the privacy of data. By making simple measurements and avoiding measuring any user sensitive data these problems can be overcome. These issues are further discussed in section 3.4.

Another reason to use a different approach to modelling network traffic is that, existing and emerging traffic modelling techniques (section 2.4, 2.6 and 2.8) rely heavily on precise mathematical analysis of extensive amounts of data. With the advent of high speed broadband networks, gathering the acceptable quantity of data needed for precise representation of traffic is a difficult, time consuming, expensive and in some cases, almost an impossible task. For example, in an ATM network with 53 bytes cell length and with a transmission rate of 1 Gbps, a router has at most 424 nanoseconds to take a routing decision and to forward it to an appropriate interface. Within current processing speeds and facilities this is difficult. The Norros process (equation (2.20)), fractional Brownian motion (section 2.6.2) and chaotic maps (section 2.6.1) - all require a huge amount of mathematical computation. These are making mathematical (and statistical) models unattractive.

Most of the existing traffic models depend on the Hurst parameter in some way. Estimation of H depends largely on the estimation method used [73, 110] and the estimated value of H of a traffic trace can vary by as much as 33% [25]. The natural way to estimate H is the variance time plot that requires curve fitting (section 2.5.2.1). This indicates that estimation of H may be vague. In other words, to date, most existing models that rely on the Hurst parameter are based on a parameter estimation method which can be imperfect or imprecise.

Mathematical and statistical models are not the only tool to deal with imperfect or imprecise data. In science and technology, imprecise or uncertain ideas have generally been handled with probabilistic models for hundreds of years. But the difficulty with accurately estimating H makes these approaches inappropriate. An alternate tool for dealing with imprecise data is fuzzy logic [88, 130]. Fuzzy logic based models can be valuable for representing processes characterized by imprecise data. Their effectiveness has been demonstrated in many industrial applications [9, 117].

Both fuzzy logic and probability theory are capable tools for dealing with imperfection. But they have some differences. Fuzzy logic is used mainly for processing and representation of vague data (ill-defined, fuzzy). Probability theory is mainly used for processing and representation of uncertainty (randomness). In probability theory, $P(A)$ is the probability that an ill-known variable x ranging on U hits the fixed well-known set A. On the other hand, when it fits fuzzy logic computes the membership of a well-known variable x ranging on U hits the ill-known set A. Fuzzy logic uses imprecise statements such as "packet arrival rate is high". It can be noted that, in this statement the variable "packet arrival rate" can be measured precisely and hence is well-known. The set "high" can not be defined precisely as it varies from network to network. Therefore, fuzzy logic based models appear to be more appropriate to represent various states of a traffic stream.

Another difference between probability theory and fuzzy logic is that probability is measured before an event occurring. In fuzzy logic, membership of a variable in a set is computed after the event has occurred. Therefore, in developing a traffic model using a measured data approach, fuzzy logic appears to be more appropriate than probabilistic models.

One of the aims of our work is to develop a router buffer management scheme. An efficient router buffer management scheme requires the prediction of a congestion state and sending some sort of control signal to the sender. In other words, a router buffer management scheme can be thought of as a control system. Fuzzy logic based models have been applied in control engineering with considerable success [60, 100, 117, 128]. Their main strength is adaptability and simplicity.

Another motivation for using fuzzy logic for modelling network traffic is its ability to reduce mathematical complexity. By means of fuzzification (section 4.1.2), a fuzzy logic system converts a large range of values into a small number of categories. This greatly reduces the mathematical complexity. For example, traffic arrival rate in a router can range from 0 to a large value. By fuzzifying, this large range of values can be classified into "small", "medium" and "large". This reduces the number of values which need to be considered by the router to just three categories. In Chapter 4 fuzzy logic and fuzzy logic based systems are reviewed and in Chapter 5 fuzzy logic based traffic models are developed. The applications of the models are discussed in Chapters 6 and 7.

Observation of self-similarity on network traffic implies that network traffic is bursty. The burstiness of a traffic stream may have significant effect on the performance of a system (such as a router) to which the stream is offered. Existing traffic models do not provide any mathematical measure of burstiness of a traffic stream. One of the aims of this research, discussed in the next section is to provide such a measure of burstiness of a traffic stream.

2.10 Measuring Burstiness of Network Traffic

In self-similar traffic, burstiness is characterized by the facts that there is no natural length for the bursts and that a burst can be within another burst, known as *burst-within-burst* [126]. This implies that, for self-similar traffic, traditional measures of burstiness such as peak-to-mean ratio or the coefficient or variation (of packet inter arrival time) [64] are almost useless. This is because for a particular value of the peak-to-mean ratio (or coefficient of variation), the burst can have practically any value of peak or mean as long as the ratio is conserved. For example, let us consider two traffic streams {2, 4, 6, 3, 0} and {3, 1, 8, 3, 5}. In both cases, peak to mean ratio is 2. But obviously the second stream is burstier than the first.

An alternative measure of burstiness is to use the H parameter . However, estimating the value of the H of a real traffic trace is difficult (section 2.5.2.1). Furthermore, the H parameter is a property only of self-similar processes. Hence the recent discovery of non-stationarity in Internet traffic has made the H parameter an even weaker candidate as a measure of burstiness.

One possible choice to measure the burstiness of a traffic stream is to apply the wavelet based Multi Resolution Analysis (MRA) energy plots developed by Abry and Veitch [3] to the traffic stream. These energy plots can be a good measure of the burstiness of a traffic stream. Their accuracy has been demonstrated by many researchers [51, 70]. Wavelet based MRA energy plots are reviewed below.

2.10.1 Energy Plots

Burstiness of a traffic stream means a traffic count such as packet counts or byte counts exhibits statistical variability at a given scale. However, burstiness can be characterized by wavelet based MRA (Multi Resolution Analysis) *energy plots* developed by Abry and Veitch [3]. Their method uses a wavelet decomposition [20] of a time series. In this method a traffic stream is treated as a counting process. In this method, the counting process at a time scale $T_j = 2^j T_0 \, (j = 0, 1, \cdots)$ is a WSS time series $X_j = \{X_{j,0}, X_{j,1}, \cdots, X_{j,n}\}$, where $X_{j,k}$ is the byte count in the kth time interval $t_{j,k}$ of duration T_j and T_0 is the reference time scale, which is the minimum interval in which counts are measured. The traffic series is decomposed into corresponding wavelet coefficients. An MRA energy plot shows the variance of the wavelet coefficients of X_j as a function of j.

The simplest form of wavelet is the Haar wavelet [33]. The Haar wavelet coefficients of X_j at scale j are defined as

$$W_{j,k} = 2^{-j/2} \left(X_{j-1,2k} - X_{j-1,2k+1} \right) \tag{2.24}$$

The energy ψ_j of X_j at scale j is defined as the variance of its wavelet coefficients,

$$\begin{aligned} \psi_j &= Var\left[W_{j,k}\right] \\ &= 2^{-j}E\left[\left(X_{j-1,2k} - X_{j-1,2k+1}\right)^2\right] \\ &= 2^{-j}Var\left[\Delta X_{j-1,k}\right] \end{aligned} \tag{2.25}$$

where, $\Delta X_{j-1,k} = X_{j-1,2k} - X_{j-1,2k+1}$.

In equation (2.25), there are an infinite number of wavelet coefficients. In practice, the energy is computed from a finite N_j coefficients. That is,

$$\psi_j \approx \frac{2^{-j}}{N_j}\sum_{k=1}^{N_j}\left(\Delta X_{j-1,k}\right)^2 \tag{2.26}$$

If X_j is a Poisson process, due to the memoryless property it is independent at any scale j. Therefore, for the Poisson process,

$$\begin{aligned} Var\left[\Delta X_{j-1,k}\right] &= 2Var\left[\Delta X_{j-1}\right] \\ &= 2^jVar\left[X_0\right] \end{aligned} \tag{2.27}$$

For a Poisson process, $Var\left[X_0\right] = \lambda T_0$. Therefore, using equation (2.25), the energy of a Poisson process of rate λ at scale j is given by,

$$\zeta_j = 2^{-j}2^jVar\left[X_0\right] = \lambda T_0 \tag{2.28}$$

Equation (2.28) shows that the energy of a Poisson process is constant and does not depend on j. Therefore, comparing the energy plot of traffic being measured to the energy plot of a Poisson process of the same arrival rate as the traffic, gives an estimate of the energy of the traffic.

Energy plots have the advantage that they do not require a huge data set and they can measure the burstiness at any time scale.

Although energy plots can be a good measure of burstiness, there is no suitable mathematical quantity to measure the burstiness. One of the aims of this research is to develop such a mathematical quantity that will not require a huge amount of data or computation.

2.11 Conclusion

This chapter reviewed some existing traffic models. Starting from the formation of the Internet, some of the characteristics of the Internet and Internet traffic from its earliest inception to current models were discussed. Early traffic models followed Poisson based memoryless models ignoring the existence of burstiness at large time scales. Eventually they became obsolete. Self-similar traffic models can capture burstiness. But they are appropriate only under some strict conditions (section 2.6.3). Observation of the existence of pervasive nonstationarity implies that self-similar models cannot be used to model Internet traffic without proper modification. Recent traffic models are more accurate but they rely on the Hurst parameter, estimation of which requires a huge amount of sample data and complex computation. Without adequate efficient models to determine appropriate traffic management parameters, we may see unstable Internet performance, overall low throughput and hence poor performance. The aim of this research is to develop a traffic model that is simple to implement and still useful in traffic analysis and performance improvement. Chapter 3 discusses some performance optimization issues and objectives.

Chapter 3

Network Performance and Management

This chapter discusses various issues and concepts that are related to network performance. In the Internet more than one networks are connected to each other by means of a router. Routers employs queues to temporarily buffer packets if required. Performance of a network is largely dependent on the proper dimensioning and management of the router queue [6, 8]. Improper buffer setting and inefficient management can lead to an unstable router queue and the result is performance degradation. This chapter discusses some of the performance parameters and several existing buffer dimensioning and queue management techniques.

3.1 Network Performance

There are different aspects to network performance. From a simple user perspective or for many applications such as file transfer or email transfer, network performance may just be the speed of the network. However, there are other aspects of network performance. The Internet is a packet switched network, where packets may be sent along different routes, stored for a while before being forwarded, or even dropped and retransmitted. There are applications where the nature of how the data is delivered is more important than how fast it is delivered. These applications may require a specific level of service, known as "Quality of Service" (QoS). Network performance is not only a measure of speed, it also reflects the quality of service of an application as seen by the end user.

3.1.1 Reasons for Performance Degradation

Often networks do not perform to their peak capacity. Possible reasons for performance degradation are:

- **Packet Loss.** Packet loss means that one or more packets fail to reach the destination while traveling across the network. Packet loss can be congestive or non-congestive. Congestive losses occur due to an over-saturated network link or as a result of inefficient router queue management. Non-congestive loss may occur from improper network settings such as an inefficient TCP stack implementation, hardware failure or signal degradation over the network media.

- **Delay.** This is the amount of time taken by a packet to be transmitted from source to destination.

- **Jitter.** Jitter in a data communication network is unwanted abrupt variation of delay. When packets are sent via different or alternate routes, jitter can be higher than the values of jitter in normal circumstances. This type of jitter is sometimes referred to as excessive jitter.

- **Out of order packets.** TCP ensures reliable delivery of packets by assigning a sequence number to each packet. Sometimes due to taking different routes or some other reason, packets may arrive out of order at a destination. Out of order packets sometimes cause duplicate acknowledgments and thus reduce overall throughput and goodput.

- **Duplicate packets.** It can happen that a packet eventually arrives at the receiver but the sender sends the packet again due to timeout. Duplicate packets mis-utilize the network resource and hence reduce overall goodput (defined in the glossary).

Packet loss causes TCP to retransmit and thus reduces overall throughput. It may cause dropouts in voice and video. Jitter and loss are the main problems that affect performance of real-time applications. Excessive jitter in such an application can cause a number of packets to be treated as lost causing dropouts and audio-video problems. Applications can be designed to cope with delay, but it is difficult to avoid jitter [121]. A recent study showed that jitter caused more problems than loss in a Voice Over Internet Protocol (VOIP) session [13].

Out of order packets may cause more problems than lost packets especially in streaming audio/video applications [13]. For data-traffic, such as e-mail or file transfer, packet reordering is not a major problem, since the data is not delivered or used until all the packets have arrived. Streaming video data, particularly MPEG video packets, have precisely defined structure, sequence and timing [76]. This timing and structure are required by the decoder to determine correctly where and when each pixel needs to be placed in the picture frame. If packets arrive out of order, they must be ordered in the right sequence before any image can be produced. This can cause video streams to stop momentarily.

3.1.2 Performance Objectives

Ideally a network should operate at its full capacity with no loss, no congestion and no queuing delay. That is, if r, ρ, q *and* v are the packet loss rate, utilization of the link, average router queue size and variance of router queuing time respectively, ideally we want ρ to be 100% and $r = q = v = 0$. This is the extreme ideal scenario and it is impossible to achieve in real networks, even in over-provisioned links. To maintain full utilization with a small buffer size, the packet loss rate may be high. This may be unacceptable for certain applications such as a Voice Over Internet Protocol session. On the other hand, to keep the loss rate to a minimum with full utilization, might require a large buffer. This may lead to an unacceptable values of v and/or q. Higher values of q can cause excessive delay and can make interactive applications unusable. Higher values of v can cause excessive jitter, making audio/video applications unusable.

Router design and setting up router buffer parameters have long been a critical issue [35, 72]. Setting up router buffer parameters is intimately related to the desired performance objectives. These objectives can be, and should be, different from network to network. For example, for a data network, maximum utilization may be a major performance objective. On the other hand, for an interactive session, small queuing delay and no packet loss are required. This research work classifies the performance objectives that can be achieved realistically into the following categories:

1. **Full utilization, small buffer size** The objective is to be close to 100% utilization with a small buffer size and hence small variance in the queuing delay. In this case there will be no control on the packet loss rate. That is, utilization of close to 100% and small buffer size is achieved at the expense

of high loss rate. Sometimes a threshold for the loss rate is set and this goal is achieved, compromising the utilization. But the main objective of this category is to have high utilization generally, this is the objective of data networks.

2. **Full utilization, small loss rate** The objective is to be close to 100% utilization with a small packet loss rate. This is achieved at the expense of a large buffer and a high variance of queuing delay. Data networks where queuing delay is not an issue might aim to meet this objective. For example, in the case of email transfer or for bulk file transfer, queuing delay is not a significant issue.

3. **Small packet loss rate, small buffer size** Time critical applications such as VOIP or streaming video normally aim for this objective. These applications cannot afford high loss rate, large buffer sizes or high jitter. However in order to maintain the quality of these applications, utilization may drop to a low value.

Other performance objectives can be specific to the application level. Examples of such objectives can be a high TCP throughput or a small end to end delay.

3.2 Buffer Dimensioning

Buffers in routers are a valuable resource. Determining the proper size of a buffer is required for the smooth flow of packets and hence is critical to overall router performance. Proper sizing of a buffer for optimum performance is known as buffer dimensioning and is a complex problem [6, 8, 35]. No single router parameter setting can be acceptable for all possible performance objectives [35]. Several approaches have been made to achieve acceptable solutions for sizing the router buffer. The most notable of them are: the Bandwidth Delay Product (BDP) model [120], the Stanford model [35] and the Buffer Sizing for Congested Link (BSCL) scheme [36].

The BDP model states that, to get optimum performance from a buffer, the buffer size should be equal to the product of the capacity of the link that can be congested and the round trip time (RTT) of the TCP connection that can be bottlenecked at that link. That is, if T is the RTT and C is the capacity of the link, the required buffer size β is given by,

$$\beta = CT \tag{3.1}$$

This model is regarded as the "rule of thumb" in buffer sizing literature [120]. It was proposed in 1994 and was widely accepted by researchers until recently when it was challenged strongly by the Stanford Model [35].

The Stanford Model [35] claims that, full link utilization can be achieved with a smaller buffer than in equation (3.1). The model states that the buffer requirement at a congested link decreases with the square root of the number of active "long" TCP flows, N. A flow is classified as "long" if it leaves the TCP slow-start process [4]. According to this model, the buffer requirement to achieve close to full utilization is given by,

$$\beta = \frac{CT}{\sqrt{N}} \tag{3.2}$$

Here it can be noted that for $N = 1$, this leads to the rule-of-thumb in equation (3.1). The main objective of this model is to maintain close to 100% utilization. But Dhamdhere and Dovrolis in [35] showed that this can lead to a very high loss rate which might not be acceptable to applications such as a VOIP session, streaming video or other interactive applications. Loss rate is a crucial TCP/IP performance metric and should not be ignored in any buffer parameter tuning scheme.

The BSCL performance objective is to achieve full link utilization while setting a threshold for the maximum loss rate. This model suggests that the buffer size should be increased proportionally to the increase in the number of active long TCP flows. The main drawback of this model is that even if it limits the loss rate and achieves high utilization, it may require a large buffer space especially when N is large. This large value of q may lead to a long queuing delay, especially in low capacity links and hence might not be acceptable to many applications. It may also increase the value of RTT and hence mislead other TCP functions. For example, if RTT is large, a TCP application may think that the packet is lost and subsequently retransmit the packet. Thus packet duplication may occur.

Another approach [35] is to set up a constraint on the maximum queuing delay at a target link. For example, if we set the maximum queuing delay constraint to 10 milliseconds for an OC-12 link ($C = 620$Mbps), the buffer space for the link should be approximately 6.2Mb. However, this may lead to a low utilization, especially in the presence of long TCP flows [35].

3.3 Queue Management and Fairness

Performance of a TCP/IP network may depend largely on the router queue management. Queue management algorithms (or simply queue management) are the algorithms that establish and manage the router queue. The module in the router that does the queue management is known as the Queue Manager (QM). Queue Managers control the length of the queue by dropping packets if required. The main tasks of a QM are:

- to establish a router queue;
- to insert packets into the queue (if the queue is not full);
- to discard packets (if required or if the queue is full);
- to extract and dispatch packets from the queue; and
- to manage the queue.

Packet drops waste network resources by means of retransmission. QM schemes can use packet marking instead of packet dropping. Packet marking notifies the participating hosts about current congestion. This scheme is known as Explicit Congestion Notification (ECN) [106]. ECN can be used only if both the communicating parties are ECN capable and agree to use ECN. With ECN, an extra bit, named an ECN bit is placed in the IP header. When a router receives an ECN capable packet and anticipates congestion, it sets the ECN bit. Thus the router notifies the sender to slow down the transmission rate by decreasing its window size.

The selection of packets for dropping/marking may have significant impact on individual TCP performance. If packets to be dropped are chosen randomly, it can happen that the packets are not picked uniformly from the available flows thus making the dropping scheme "unfair". *Fairness* is the criterion that ensures uniform distribution of available resources among the available flows. Fairness in a router can be measured by the Fairness Index, *FI* [66], defined as:

$$FI = \frac{\left(\sum_{i=1}^{N} throughput_i\right)^2}{N\sum_{i=1}^{N} throughput_i^2} \tag{3.3}$$

where, $throughput_i$ is the throughput of the *ith* flow and N is the total number of active flows.

The primary goal of a QM is to meet its performance objectives without sacrificing fairness. Queue management can be classified into two categories: Passive Queue Management (PQM) and Active Queue Management (AQM).

3.3.1 Passive Queue Management

Passive Queue Management (PQM) schemes do not do any preventative packet drops until the buffer is full. Once the buffer is full, all the arriving packets are dropped. Therefore, PQM can either be in a "no packet drop" state or a "100% packet drop" state. It does not take any congestion avoidance steps nor does it send any congestion warnings. An example of PQM is the drop tail scheme where packets are dropped from the tail of the queue.

3.3.2 Active Queue Management

Active Queue Management (AQM) takes preventive measures to avoid congestion. These schemes perform packet drops before the queue is full. Generally the packet drop rate increases as the queue length increases. Preventive packet drops provide an implicit feedback mechanism to notify senders of the congestion state. Many AQM algorithms have been proposed [57]. A few of the more notable ones are discussed here.

3.3.2.1 Random Early Detection

The default AQM scheme recommended by the Internet Engineering Task Force (IETF) is Random Early Detection (RED) [47]. A router implementing RED does not drop any packets until the queue length reaches MIN_{th} and when the queue reaches MAX_{th}, it drops all packets. When the queue size is in between these two thresholds, the drop probability is linearly proportional to the queue length. At time t, the average queue length $\bar{q}(t)$ is calculated as

$$\bar{q}(t) = (1 - w)\bar{q}(t-1) + wq(t) \tag{3.4}$$

where w is the weight parameter with $0 < w < 1$. The packet drop probability p is calculated as:

$$p = MAX_{drop}\frac{\bar{q}(t) - MIN_{th}}{MAX_{th} - MIN_{th}} \tag{3.5}$$

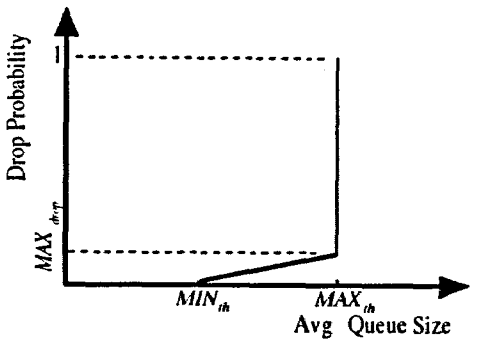


Figure 3.1: Drop Probability of Random Early Detection.

where MAX_{drop} is the maximum drop probability.

The drop function similar to equation (3.5) where the drop probability increases (or decreases) linearly is known as linear drop function. The drop probability of RED scheme is shown in Figure 3.1.

The performance of RED depends significantly on its four control parameters MIN_{th}, MAX_{th}, MAX_{drop} and w. Floyd suggests that MAX_{drop} should not exceed 2% [45, 47]. He also suggests using $MAX_{th} \sim 3MIN_{th}$ [45].

In RED, the drop probability does not depend on per-flow bandwidth consumption or buffer occupancy. Equation (3.5) shows that the drop probability p is same for all available flows. Since all the flows have the same loss rate, a flow using little bandwidth or occupying a small or no buffer space will still experience packet loss making it unfair to this type of flow. Several audio and video applications do not slow down even if a congestion is state is notified. These types of congestion non-aware flows can make RED drop packets at a high rate from all flows. Hence RED is not a fair system for different types of traffic flows.

3.3.2.2 Stabilized Random Early Detection

Random Early Detection can suffer from unstable queue length [21, 84]. *Stabilized random early detection* (SRED) makes a RED queue stable [101]. It is a variation of RED that stabilizes the queue level with a load-dependent drop probability.

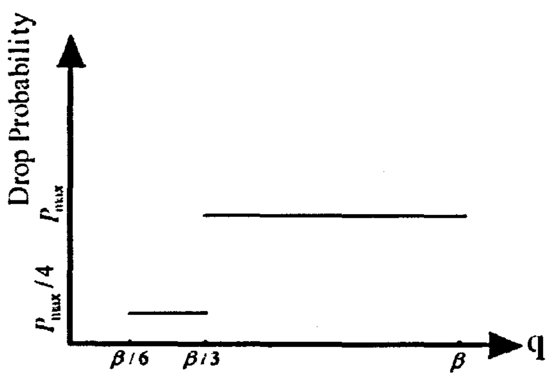


Figure 3.2: Drop Probability of Stabilized RED

Stabilized RED maintains a list of M recently served flows. On each packet arrival, the arriving packet's flow is compared with a randomly chosen flow from the list. If these two match, it is termed as a "hit", if not it is a "miss". From the recent hit-miss sequence the hit probability $P(t)$ is computed. If there are a large number of active flows, the hit probability will be small.

In SRED, different drop probabilities are used depending on the value of $P(t)$. When $P(t)$ is smaller than $\frac{1}{256}$, the drop probability is given by

$$p_{sred_1}(q) = \begin{cases} p_{max} & \text{when } \frac{\beta}{3} \leq q < \beta \\ \frac{p_{max}}{4} & \text{when } \frac{\beta}{6} \leq q < \frac{\beta}{3} \\ 0 & \text{when } 0 \leq q < \frac{\beta}{6} \end{cases} \tag{3.6}$$

where p_{max} is the maximum packet drop probability, q is the instantaneous queue size and β is the buffer capacity. Thus SRED uses a step drop function. The drop probability for SRED is shown in figure 3.2.

Equation (3.6) is used when $P(t)$ is small, i.e., when the number of active flows are large. Higher values of $P(t)$ are associated with a relatively small number of active flows. When $P(t)$ is greater than $\frac{1}{256}$, SRED uses the following drop probability

$$p_{sred_2}(q) = \frac{p_{sred_1}(q)}{65536}(N_a)^2 \tag{3.7}$$

where N_a is the number of active flows. That is, the drop probability for SRED, $p_{sred}(q)$ is given by

$$p_{sred}(q) = \begin{cases} p_{sred_1}(q) & \text{when} \quad 0 < P(t) < \frac{1}{256} \\ p_{sred_2}(q) & \text{when} \quad \frac{1}{256} \leq P(t) < 1 \end{cases} \tag{3.8}$$

Unlike RED, SRED uses a drop function that can increase (or decrease) only in steps. It has the feature that over a wide range of load levels helps it stabilize its buffer occupation at a level independent of N_a. However, like RED, SRED is still not a fair system.

3.3.2.3 Fair Random Early Detection

Fair Random Early Detection (FRED) [80] overcomes the fairness problems suffered by RED and SRED. It uses per-active-flow accounting. A drop rate for each active flow is calculated based on the buffer space currently occupied by each flow.

Fair RED introduces the parameters min_q and max_q, the minimum and the maximum number of packets each flow is allowed to buffer. The parameter min_q is calculated by dividing the average queue length by the current number of active flows. Fair RED allows a flow to buffer min_q packets without dropping. All additional packets up to max_q are subject to RED's random drop. After a flow has already buffered max_q packets, all additional packets are normally dropped.

As FRED computes per-flow drop probability based on the current buffer occupancy, it is a fair system. Like RED, FRED also uses a linear drop function.

From the above discussion it is evident that optimizing a router's performance requires addressing both buffer dimensioning and queue management. A correctly sized buffer with inefficient queue management or an efficient queue management scheme with an incorrect buffer size can lead to non-optimal router performance. In section 3.1.2, how the performance objective to be met can vary from network to network is described. The criteria that we wish to optimize (such as high utilization or loss rate) can be in conflict with each other. This motivates us in section 3.5 to define the router performance optimization problem as a Multi-objective Optimization (MO) problem. The main reason for formulating the router performance optimization problem in this way is that there exists a rich set of solution methodologies to solve such problems. These solution methodologies are discussed in Chapter 4.

In the next section network traffic measurement is reviewed, which is a methodology for collecting and analyzing traffic data. Network designers use the collected data for network performance analysis or to validate a traffic model. Evaluation of a traffic model or a router performance optimization scheme also requires measurement of traffic data.

3.4 Network Traffic Measurement

Network measurement refers to the measurement of traffic parameters such as the number of bytes transmitted, the number of packets transmitted or link utilization. Network measurement is the primary requirement for network monitoring and network behavior analysis. Network measurement can be classified in several different ways. Most monitoring and behavior analysis requires the measurement of packet headers only. Some analysis requires inspection of the user data portion of a packet. Based on the content of measurement, network measurement can be of two types: content non-aware measurement and content-aware measurement. Based on the mode of measurement, network measurement can be classified into two types: active and passive.

3.4.1 Content Non-aware Measurement

Most traditional network applications (such as telnet, WWW, FTP and email) use predefined TCP or UDP ports. These predefined port numbers are called *well-known* port numbers. Well-known port numbers are assigned and maintained by the Internet Assigned Numbers Authority (IANA). Most traditional traffic measurement tools use port based recognition and classification. They can be referred to as content non-aware traffic measurement system. They are also known as *port based measurements.* Content non-aware tools can be classified as:

- packet based; for example TCPDUMP [62] and Ethereal [29]; and

- flow based; for example Cisco's Netflow [23] and CAIDA's CoralReef [23].

3.4.2 Content-aware Measurement

Many modern applications use either dynamic port allocation or use other protocols (such as HTTP) as a wrapper to go through *firewalls* and *filters.* A wrapper is a software program that contains other software or data, so that the contained elements are disguised or hidden. An application can use non standard port numbers because:

- some applications may want to bypass port based access rules and filters;
- use of Network Address Translation (NAT) and port forwarding;
- non-privileged or ordinary users are often restricted to use high port numbers (usually > 1024); and
- some protocols are designed to use dynamic port allocations that cannot be known in advance (for example: Real Time Protocol, Passive File Transfer Protocol).

Port based identification can lead to inaccurate identification of an application. As a result, content-aware methods, such as Cisco's NBAR [23] have become more popular. Content-aware measurement tools capture both packet header and payload.

Although these methods can identify applications and capture traffic characteristics with higher precision, they require inspection of the payload [18, 83]. As the payload of a packet contains user data, it might contain private and sensitive information. Also content-aware measurements require more computing power than content-nonaware measurements [85].

3.4.3 Active vs Passive Measurement

Network measurement can be active or passive. In active network measurement, the measurement system generates and injects packets to measure a network characteristic. Examples of such approaches are the *ping* and the *traceroute* utilities [87]. Ping is used to estimate network latency or round trip time of a particular destination. The traceroute utility is used to determine network routing paths.

Passive measurement does not inject any packet to the network. A passive network measurement or monitoring tool observes and records the packet traffic data. Examples of such measurement tools are NeTraMet [10] and NG-MON [56].

Active measurement uses network bandwidth as it injects packets to the network. Active measurement can also be time consuming as it requires a packet to travel over a network. Another drawback of active measurements is that they can be sensitive to network performance and security [91]. Therefore, in the proposed traffic model and router queue management scheme only passive measurements should be used.

3.5 Router Performance Optimization: A Multi-Objective Optimization Problem

This section will show that a router performance optimization problem can be expressed as a Multi-Objective optimization (MO) problem. In section 3.5.1 the definition of an MO problem is reviewed and in section 3.5.2 the router optimization problem is formulated as an MO problem.

3.5.1 Multi-Objective Optimization Problems

The optimization of a function with more than one (possibly conflicting) objective, it is called a multi-objective optimization problem (MO Problem or MOP) [48]. Formally, a MO problem with n decision variables and M objectives is defined as [34]:

$$\begin{array}{lll} \text{minimize} & f_m(\mathbf{x}), & m = 1, 2, \cdots, M \\ \text{subject to} & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \cdots, J \\ & h_k(\mathbf{x}) = 0 & k = 1, 2, \cdots, K \\ & x_i^{(L)} \leq x_i \leq x_i^{(B)} & i = 1, 2, \cdots, n \end{array} \tag{3.9}$$

where the solution $\mathbf{x}^* = (x_1, x_2, \cdots, x_n)^T$ is a vector of n decision variables. The functions $g_j(\mathbf{x}) \geq 0$ are J inequality constraints and $h_k(\mathbf{x}) = 0$ are K equality constraints. The last set of constraints, the variable bounds, restrict each decision variable x_i to take a value in between $x_i^{(L)}$ and $x_i^{(B)}$.

In equation (3.9), $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_M(\mathbf{x}))^T$ are the M objective functions. A solution $\mathbf{x}$ that satisfies all the $(J + K)$ constraints and all of the $2n$ variable bounds (n lower bounds and n upper bounds) is called a *feasible* solution. If a solution does not satisfy all of these constraints and variable bounds, it is called an *infeasible* solution.

In equation (3.9), the minimization problem can be transformed to a maximization one by multiplying the objective function by -1 [34]. Therefore, the objective functions can be of mixed type.

3.5.2 Router Performance Optimization as an MO Problem

The router performance optimization may be formulated as a multi-objective optimization problem because it involves optimizing a function of more than one (conflicting) objectives. For example, we want to achieve close to full utilization of a link with small queues and minimal packet drops. Minimal packet drops tend to increase the queue length in a congested link. Similarly, higher utilization may be accompanied by longer queues.

Let
ρ be the utilization of the link
l be the average packet drop rate
q be the average queue size
v be the queuing time variance.

An AQM scheme requires optimization of the following objectives:

- maximize ρ (or minimize $-\rho$)
- minimize l
- minimize q
- minimize v.

In this MO problem, the decision variables ρ, l, q and v can be functions of other variables. In the case of RED, these decision variables are functions of model parameters MIN_{th}, MAX_{th}, MAX_{drop} and w. Therefore, in the case of

RED, a feasible solution is to find the values of these parameters so that the four objectives above are met. Chapter 4 discusses existing methodologies to find such a feasible solution.

Performance of a router depends on both the buffer dimensioning and queue management scheme. In section 3.2 and 3.3 it has been reviewed that, historically these two problems have been treated separately. A buffer dimensioning method does not determine the queue management scheme that should be used, nor does an AQM scheme determine the optimum buffer size. This can make a buffer management scheme inefficient. For example, the BDP model in equation (3.1) or the BCSL scheme specify a larger buffer than the Stanford model (section 3.2). In section 3.3.2.2, it was mentioned that SRED starts dropping packets when the queue size reaches $\frac{\beta}{6}$. Therefore, if a router with a buffer size equal to the size specified by BDP or BCSL uses SRED, a large portion of the buffer will be under-utilized. Similarly, if a router with a small buffer size (as specified by the Stanford model) uses drop-tail, the buffer might become full after buffering few packets. Both these cases may lead to inefficient management of a router buffer.

Performance objectives (section 3.1.2) are addressed in the buffer dimensioning problem. This work argues that the objectives should also be addressed in the queue management scheme. Buffer dimensioning is implemented in router hardware. At the time of its manufacture, the performance objectives of the network for the router are unknown. On the other hand, AQM is implemented in the router by means of software. The parameters of an AQM implementation in a router can be adjusted. Thus addressing the performance objectives in the queue management scheme makes a router's performance flexible, adaptive and scalable. The next section addresses the design principles of the queue management scheme which is proposed in Chapter 7.

3.6 Design Principles for a Queue Management Scheme

As argued above performance objectives should be addressed in the queue management. Ideally an ideal queue management scheme should be adaptive so that any performance goal can be met and the performance objectives should be met in the queue management scheme rather than in buffer dimensioning. Therefore, the first design principle for the router management scheme is **adaptability.** We want a queue management scheme that is adaptive so that any performance objective can be met by adjusting a minimal number of parameters.

The next design principle for our queue management scheme is **stability**. In this context, the design choice is whether to use a linear drop function or a step drop function. Most AQM schemes use a linear drop function (section 3.3.2). These drop functions are dependent on buffer capacity. The linear drop function that is used in the original RED can make the queue unstable. SRED keeps the queue stable by using a step drop function (section 3.3.2). Hence it is decided to use a step drop function rather than a linear drop function.

Fairness (section 3.3) is another issue that has to be addressed by any AQM scheme. Both the original RED and SRED are not optimally fair. FRED (section 3.3.2) approaches fairness by selecting the flows for packet dropping according to their proportion of buffer occupancy. It can achieve fairness, but the number of packets dropped might not be the minimum. To illustrate, let us consider the case where two flows, one with a constant arrival rate and the other with a continuously increasing arrival rate are occupying equal amounts of buffer space. To maintain a small queue, if two packets need to be dropped, it is likely that FRED will drop one packet from each flow. This affects the performance of both the flows. The same level of queue length might be maintained by dropping one packet from the second flow at an earlier stage. In that case, the second flow would have slowed down earlier. Thus a similar queue level might be maintained by dropping one packet instead of two. Therefore, our design goal is to use some other suitable traffic characteristics to select a flow for packet dropping so that **fairness is achieved by dropping the minimum number of packets.**

In section 3.3.2.3, it is showed that FRED uses per-flow accounting to make it fair. Another design principle of our queue management scheme is, if it has to use per-flow accounting it will be restricted to **content-nonaware passive measurements** only.

3.7 Conclusion

This chapter discussed some key technologies for performance and measurement in computer networking. The in-built flow control mechanism of TCP can lead to network congestion. Routers use many queue management techniques to avoid such congestion. This chapter highlighted several AQM algorithms (RED, SRED and FRED). This chapter also discussed content-aware and content-nonaware measurement technologies.

This chapter formulated router performance optimization as an MO problem. The design principle of an efficient queue management scheme was also discussed. In theory, a router can improve the overall QoS by inspecting the payload of the packets using some content-aware measurement techniques. But due to privacy, security and performance issues, this cannot be followed by routers in many situations. Active measurements waste network resources. Therefore, ideally a router queue management scheme should achieve its performance objectives along with fairness and QoS without using content-aware or active measurement. This is one of the main aims of this research. Soft computing optimization algorithms can be useful in this context. In section 2.9 motivations for developing a fuzzy logic based traffic model is stated. Chapter 4 reviews fuzzy logic and discusses the available tools to deal with an MO problem.

Chapter 4

Soft Computing

Chapter 2 discussed the motivation for developing a fuzzy logic based traffic model (section 2.9). Fuzzy logic is a branch of Soft Computing (SC). Soft Computing [129] resembles human reasoning. It is tolerant of uncertainty, approximation and partial truth. Soft Computing exploits this tolerance to produce robust solutions to different types of complex problems including modelling and analysis of complex phenomena. Soft Computing is a collection of such tolerant, adaptive and flexible computational techniques including fuzzy logic and evolutionary computing. The first part of this chapter reviews fuzzy logic and a fuzzy logic system. In Chapter 3 the router optimization problem was formulated as a multi-objective optimization problem. The second part of this chapter reviews available approaches and, in particular, evolutionary computing approaches to solving MO problems.

4.1 Fuzzy Logic

Fuzzy logic (FL) is a branch of mathematics conceived by Zadeh [132]. He argued that mathematically precise input is not needed to maintain acceptable control of a system. A fuzzy logic system can accept imprecise data and still be capable of producing highly adaptive solutions. It is a simple but powerful way to analyze and control complex systems. With fuzzy logic a statement (proposition) is represented with degrees of "truthfulness". For example, the statement "today is hot", can be 100% true (i.e., truthfulness is 100%) if the temperature is more than 35 degrees, can be 70% true (i.e., truthfulness is 70%) if the temperature is 30 degrees, and 0% true (i.e., truthfulness is 0%) if the temperature is -2 degrees. Much literature [93, 122] exists on FL systems and FL control systems. All the definitions in this chapter are written following [93] and [131].

4.1.1 Fuzzy Sets, Membership Functions and Linguistic Variables

Mathematically, a fuzzy set F is a mapping of a domain to $[0, 1]$. The domain is called the *Universe of Discourse* and is denoted by U. For example, it may be that the temperature of a place cannot be lower than -70 degrees or higher than 60 degrees. Therefore, the universe of discourse for the temperature of that place is the range [-70, 60]. The mapped value is called the *grade of membership* or the *degree of membership* and is denoted by μ_F. Unlike conventional sets (known as crisp sets), the μ_F can have any value between 0 and 1. Let us consider the fuzzy sets Cold, Warm and Hot. A temperature of 27 degrees can be warm to some people and can be hot to others. In fuzzy logic, the variable temperature can be a member of both Warm and Hot fuzzy sets with corresponding grades of membership. It can be said that a temperature of 27 degrees is warm with 60% "truthfulness" and is hot with 40% "truthfulness". This is equivalent to saying, temperature is Warm with degree of membership 0.4 and is Hot with degree of membership 0.6. In the "crisp" case (or in ordinary Boolean logic), a temperature of 27 degree can either be warm or hot.

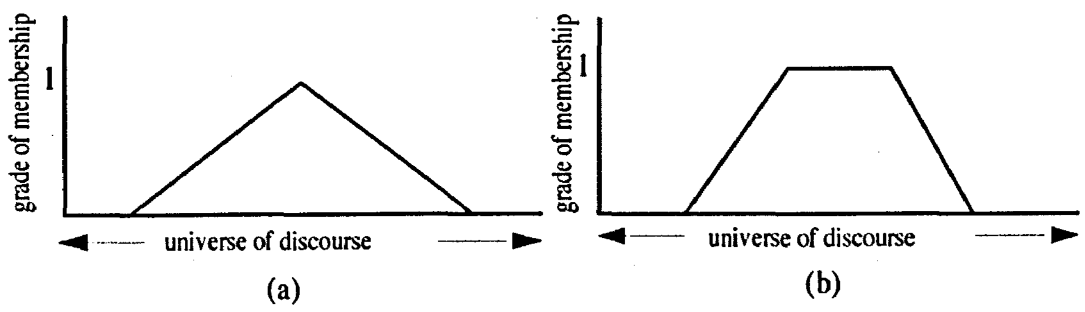


Figure 4.1: Two Popular Shapes of Membership Functions. (a) Triangular (b) Trapezoidal

The variable temperature can have any value in U (-70 to 60 degrees) and can be a member of more than one fuzzy set (e.g. both Warm and Hot) at the same time with corresponding grades of membership (0.4 and 0.6 respectively). Such a variable is called a linguistic variable.

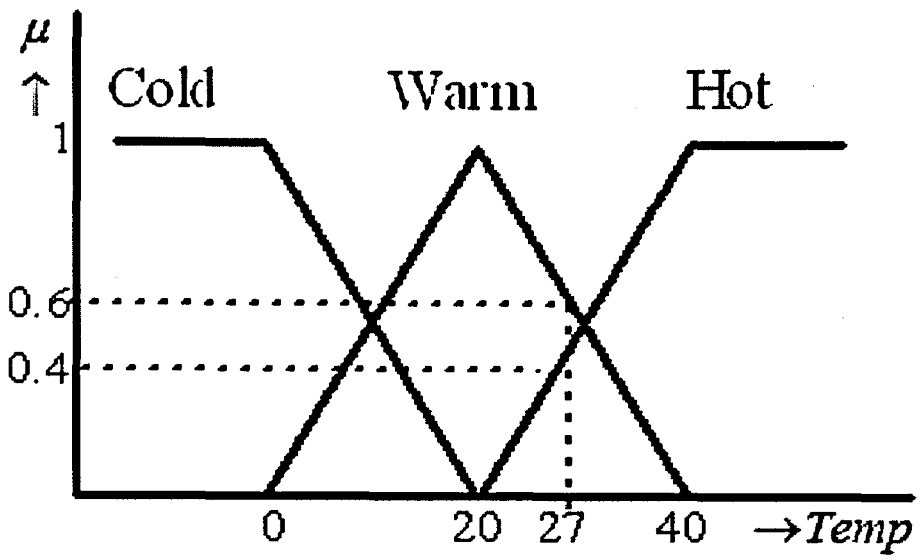


Figure 4.2: Fuzzy Sets Cold, Warm and Hot with Triangular Membership Function

Often the grade of membership of such a linguistic variable of a fuzzy set is represented by a suitable mathematical function. This function is called a *membership function.* In other words, a membership function is a graphical (or tabulated numeric) function that assigns membership values between 0 and 1 to the crisp values of an input variable over its universe of discourse. The most popular choices for the shape of membership function are triangular and trapezoidal as shown in Figure 4.1. Perhaps the simplest one is the triangular one. Figure 4.2 shows how the crisp value of 27 degree is mapped to a fuzzy domain using a triangular membership function.

The *support* of a fuzzy set F is the crisp set of all values in U such that $\mu_F(x) > 0$. In Figure 4.2, the Warm fuzzy set can have non-zero membership value if the temperature is between 0 and 40 degrees. Therefore, the support of fuzzy set Warm is 0 to 40 degrees. If the support of a fuzzy set is a single point x' in U and $\mu_F(x') = 1$, it is called a *fuzzy singleton.*

The *core* of a fuzzy set F is the crisp set of all values in U such that $\mu_F(x) = 1$. In Figure 4.2, the core of the Warm fuzzy set is 20 degrees.

4.1.2 Fuzzy Logic Systems

A fuzzy control system or a fuzzy logic system or a fuzzy model consists of three basic steps: fuzzification, inferencing and defuzzification [77, 127]. For a given set of input variables, the goal of a fuzzy logic system is to produce an output control variable. The structure of a fuzzy control system is shown in Figure 4.3. Fuzzification is the process of mapping from the measurable crisp input to the fuzzy sets defined in the corresponding universe of discourse. Fuzzy inference is the decision making logic. It determines the fuzzy output based on a fuzzy rule-base. Defuzzification produces crisp outputs of the control variable. The fuzzy logic system of Figure 4.3 is described below.

4.1.2.1 Fuzzification

In a fuzzy logic system, the inputs are given as crisp values. In order to make them usable in a fuzzy logic system we need to *fuzzify* them. Fuzzification assigns linguistic values to a variable using a (relatively small) number of membership functions. Preprocessing of a large range of values into a small number of fuzzy categories greatly reduces the computational complexities for a given problem.

There can be at least two types of fuzzification method: singleton and non-singleton [93]. The simplest and the most commonly used fuzzifier is the *singleton* fuzzifier which is essentially a singleton (section 4.1.1). In a singleton fuzzification, the fuzzifier maps a crisp input point to a fuzzy singleton. If A' is a fuzzy singleton with support x', then $\mu_{A'}(x') = 1$ for $x = x'$ and $\mu_{A'}(x') = 0$ for $x \neq x'$.

The non-singleton fuzzifier [97] assumes that the inputs are corrupted by noise. Each input crisp value is treated as a fuzzy number and hence has a corresponding membership function. Conceptually, the non-singleton fuzzifier implies that the given input value x' is the most likely value to be the correct one. However as the input is corrupted by noise, neighboring points may also be a correct value, but to a lesser degree. In our work fuzzy logic is used to model network traffic. The input crisp values are packet or byte counts, where there is not normally any corruption by noise. Hence, in this work only the singleton fuzzification is used.

In general fuzzification requires:

1. Identification of the universe of discourse for each linguistic variable;

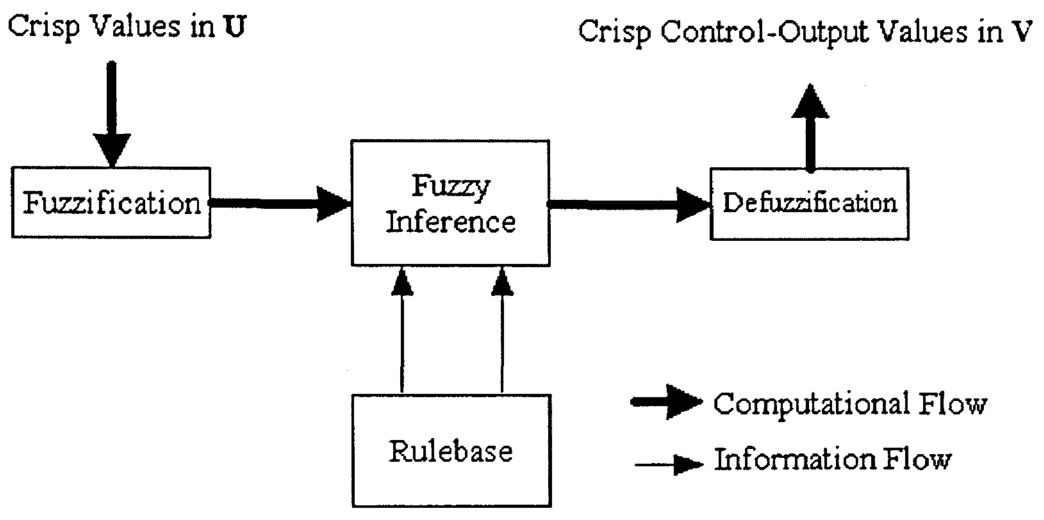


Figure 4.3: The modules in a Fuzzy Logic based Control system. The Fuzzification module maps the crisp value to fuzzy sets. The Inference Engine produces fuzzy output based on a rulebase. The defuzzification module produces crisp output.

2. Choice of input fuzzy sets;
3. Determination of support and core for each fuzzy set;
4. Choice of membership function; and
5. Fuzzification method.

In the temperature example of the previous section:

1. The universe of discourse for the temperature is identified to be -70 to 60 degrees;
2. The input fuzzy sets are Cold, Warm and Hot;
3. Support for the Warm fuzzy set is 0 to 40 degrees (<0 is Cold and >40 is Hot) and the core of the Warm fuzzy set is 20 degrees;
4. The triangular membership function is chosen. In Figure 4.2, the membership function of the fuzzy set Warm is defined to be

$$\mu_F(x) = \begin{cases} 0 & x \le 0 \\ \frac{x}{20} & 0 < x \le 20 \\ \frac{40-x}{20} & 20 < x < 40 \\ 0 & x \ge 40 \end{cases} \tag{4.1}$$

5. Singleton fuzzification is used.

4.1.2.2 Fuzzy Inference

The Fuzzy inference process does the actual decision making based on a set of rules, known as a fuzzy rulebase. The rule-base is composed of a set of if-then rules of the form

$$R^{(l)}: \text{ if } x_1 \text{ is } F_1^l \text{ and } \cdots\cdots \text{and } x_n \text{ is } F_n^l \text{ then } y \text{ is } G^l \tag{4.2}$$

where $l = 1, 2, \cdots, m$ and m is the number of rules. $x_j\,(\text{for}\, j = 1, 2, \cdots, n)$ are the inputs to the fuzzy system, y is the output variable, F_j^l are the input fuzzy sets, G^l is the output fuzzy set and n is the number of input linguistic variables. The "if" part of a rule is called an *antecedent* and the "then" part is called the *consequent* of the rule. Each "if x_j is F_j^l" part of the antecedent is known as a *term*.

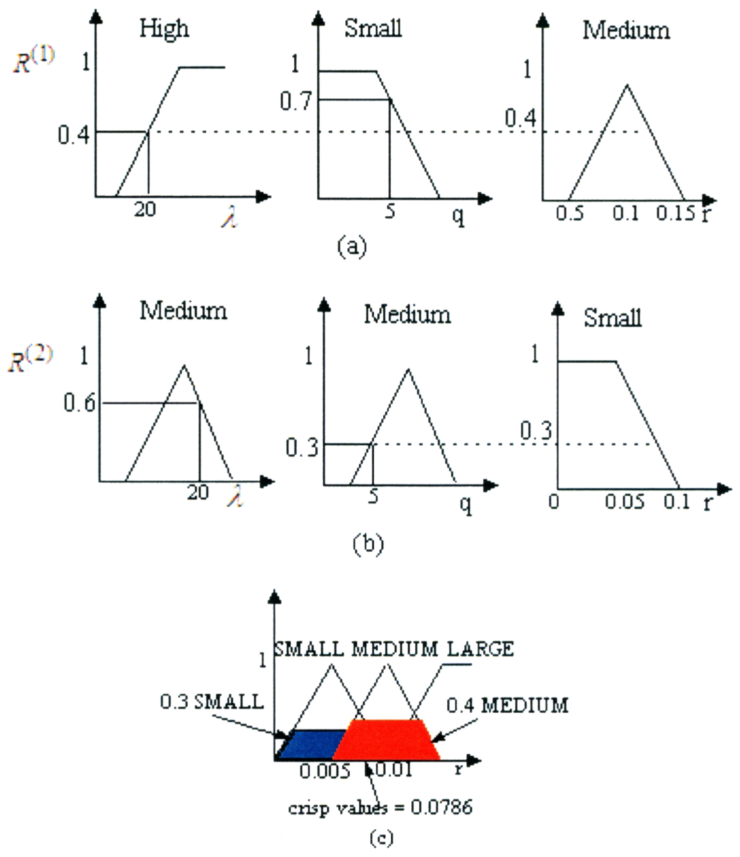


Figure 4.4: Fuzzy Inferencing and defuzzification of the Router Buffer Control Problem using min Inference and centroid defuzzification. The dashed line corresponds to the min operation, as in (a) min(0.4, 0.7)=0.4 and in (b) min(0.6, 0.3)=0.3. (c) is the defuzzification.

The inference process computes the grade of membership for each term of the antecedent. Then, the "degree is fulfillment" of the entire antecedent is computed by using a fuzzy AND, which is simply the smallest value of all the grades of membership of the n terms. Finally, the grade of membership of the consequent is computed from the grade of membership of the antecedent by using a suitable operator. The most popular operator is the *min* operator, which is simply the minimum of the membership values of all the terms of a rule.

4.1.2.3 Defuzzification

Defuzzification is the mechanism for converting the output fuzzy sets of the inference engine into nonfuzzy or crisp values. The objective is to derive a single numeric crisp value that best represents the output of the inference engine. The defuzzification method to be used depends on the entire fuzzy system design. Perhaps, the most popular defuzzifier is the *centroid defuzzifier* [19, 93], where the output of the defuzzifier is calculated as

$$\bar{y} = \frac{\left[\sum_{l=1}^{m} H_l \mu_{G^l}(y)\right]}{\left[\sum_{i=1}^{m} \mu_{G^l}(y)\right]} \tag{4.3}$$

Packet Arrival Rate (λ)	Queue Length (q)	Packet Drop Rate (r)
High	Small	Medium
Medium	Medium	Small

$R^{(1)}$: if λ is High and q is Small then r is Medium
$R^{(2)}$: if λ is Medium and q is Medium then r is Small

Figure 4.5: Inference rules for a Fuzzy Logic Controlled Router Buffer

where, m is the number of rules, $\mu_{G^l}(y)$ is the membership value of the consequent part of the l^{th} rule for y and H_l is the center of (the support of) the output fuzzy set of the l^{th} rule.

To illustrate the entire process, let us consider the fuzzy logic control system of a router queue management system. The queue manager drops an arriving packet with a drop probability r. The drop probability is calculated based on the packet arrival rate and the current queue length. The inference rules for this system are shown in Figure 4.5. Suppose at any time instant, the packet arrival rate λ is 20 and the queue length q is 5. The fuzzy logic system needs to calculate r. The steps are described below.

Step 1: Fuzzification. Suppose these values are fuzzified using the membership function of Figure 4.5, so that, the arrival rate of 20 is assigned 0.4 High and 0.6 Medium. Similarly, the queue length of 5 is assigned 0.7 Small and 0.3 Medium.

Step 2: Fuzzy Inference. The first rule $R^{(1)}$ states that, if the packet arrival rate (λ) is High and the queue length (q) is Small, then the drop rate (r) is Medium. The rule has two terms: the first term is "λ is High" and the second term is "q is Small". The first term has the membership value 0.4 (since, arrival rate is 0.4 High) and the second term has membership value 0.7 (since the queue length is 0.7 Small). Therefore, the membership value for the antecedent is $min(0.4, 0.7) = 0.4$. That is, the degree of fulfillment of rule $R^{(1)}$ is 0.4. Consequently, the output of the fuzzy inference will be 0.4 Medium, i.e., packet drop rate is Medium with membership 0.4. The fuzzy inferencing process of $R^{(1)}$ and $R^{(2)}$ are shown diagrammatically in Figure 4.4a and 4.4b respectively. In this example, the support for the Medium fuzzy set is assumed to be [0.05, 0.15] and the center of the support to be 0.1. Similarly, the support of the Small fuzzy set is assumed to be [0, 0.1] and centered at 0.05.

Step 3: Defuzzification. In this example (Figure 4.4), $m = 2$. For the first rule, the output fuzzy set (G^1) is Medium, $\mu_{G^1}(y) = 0.4$ and the center of Medium is 0.1. Similarly, for the second rule, the output fuzzy set (G^2) is Small, $\mu_{G^2}(y_2) = 0.3$ and the center of Small is 0.05. Therefore, according to the center of gravity defuzzification rule in equation (4.3), the crisp output packet drop rate is given by,

$$p = \frac{0.1 \times 0.4 + 0.05 \times 0.3}{0.4 + 0.3} = 0.0786 \tag{4.4}$$

That is, the queue management system of the example 4.5 will set a packet drop probability of 7.86%.

In section 3.5 the router performance optimization problem was formulated as a multi-objective optimization problem. Next some solution methodologies for such a problem is discussed.

4.2 Approaches to Solving MO Problems

Multi-objective optimization problems can be solved using two approaches: classical approaches and evolutionary approaches [34]. In this section these approaches are reviewed. This section also describes why the evolutionary approach is more appropriate for solving an MO problem such as the router performance optimization.

Solutions to a multi-objective optimization problem may be explained by means of "Pareto optimality". In section 4.2.1 the notion of Pareto optimality is reviewed and in sections 4.2.2 and 4.2.3, classical and evolutionary approaches to solving MO problems are reviewed.

4.2.1 Pareto Optimality

In a single objective optimization problem, the goal is to achieve the best single solution. But for an MO problem, with several possibly conflicting objectives, there may be no single optimal solution. Therefore, the solution is chosen from a set of solutions by making a "trade-off" or *compromise*. Therefore, the notion of "optimality" is different in the case of multi-objective optimization problems. This type of "compromised" optimality may be addressed by *Pareto optimality* [48].

Let $\mathcal{F}$ denote the global set of all vectors that satisfies all the constraints and the variable bounds of an MO problem in equation (3.9). A vector $\mathbf{x}^* \in \mathcal{F}$ of decision variables is *Pareto optimal* if there exists no other $\mathbf{x} \in \mathcal{F}$ such that $f_m(\mathbf{x}) \leq f_m(\mathbf{x}^*)$ for all $m = 1, 2, \cdots M$ and that $f_h(\mathbf{x}) < f_h(\mathbf{x}^*)$ for at least one $h \in m$.

Pareto optimal solutions are solutions within the search space whose corresponding vector components cannot be improved simultaneously. In other words, a solution $\mathbf{x}^* \in \mathcal{F}$ of the MO problem in equation (3.9) is Pareto optimal if there exists no feasible solution $\mathbf{x} \in \mathcal{F}$ which would decrease some objective functions without causing a simultaneous increase in at least one other objective.

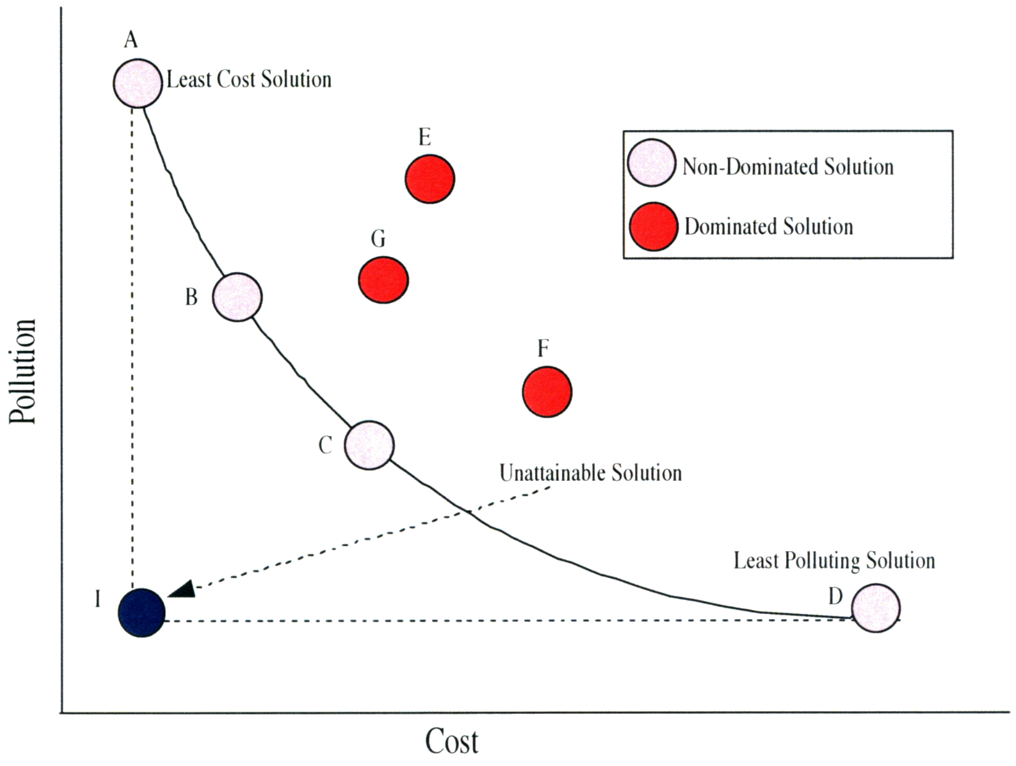


Figure 4.6: The Pareto Optimal Set for the motor manufacturer problem. The cheapest solution makes the highest Pollution and the costliest solution makes the lowest pollution. The ideal solution (I) is unattainable.

For a given MO problem, Pareto optimality can provide more than one solution. The set of solutions that satisfies the Pareto optimality is known as the *Pareto-optimal set*. All the solutions in the Pareto-optimal set are called *non-dominated* or *non-inferior* or *efficient* solutions. A solution that is not Pareto-optimal is called a *dominated* solution.

Let us consider as an example a problem of a car manufacturer. Their production record shows that it is costliest to manufacture a car that makes the least pollution and that the cheapest car makes the highest pollution. The manufacturer wants to make a low cost car that makes less pollution. This is a MO problem where the objectives are conflicting. A possible solution scenario is shown in Figure 4.6. The figure shows that the solution marked I is the ideal but unattainable solution. A, B, C and D are the Pareto-optimal solutions. The solutions E, F and G are dominated solutions.

Using the notion of Pareto optimality classical and evolutionary approaches for solving MO problems are now discussed.

4.2.2 Classical Approach

Classical multi-objective optimization methods have been proposed for the last forty years [28, 75]. Cohon in [27] classified them into the following two types:

- Generating methods
- Preference based methods

Generating methods do not use any prior knowledge of the relative importance of objectives. These methods use a heuristic approach to find a single optimum solution. At first a few non-dominated solutions are generated by observation and then the decision maker chooses one solution from the generated solutions.

In preference based methods, some preferences of the relative importance of the objectives are known. These preferences are used in the optimization process. In these methods an interaction with the Decision Maker (DM) is generally required during the optimization process.

Deb in [34] classified the classical methods for solving MO problems into the following two groups:

- Direct methods
- Gradient-based methods

In direct methods, only the constraint values and the objective function are used to guide the search method to find an optimum solution.

In the gradient based methods, first and/or second order derivatives of the objective function are also used in the search procedure.

Many researchers have shown that classical approaches are not suitable for solving MO problems [34, 90]. The main disadvantages of classical approaches, as stated in [34] are:

1. Most classical algorithms tend to become stuck in a suboptimal or dominated solution;
2. Some classical methods require interaction with the Decision Maker;

3. Convergence to an optimal solution depends on the initial generated solutions;

4. They are not efficient in dealing with MO problems that have a discrete search space; and

5. They are usually slow.

Because of these problems it is decided to use an evolutionary approach to solve the router optimization problem. But evolutionary approaches may also suffer from problems 1 and 3 mentioned above. This is further discussed in section 4.2.3.1. The next section reviews the evolutionary approach for solving an MO problem.

4.2.3 Evolutionary Approach

Evolutionary Computing (EC) [7, 59] is a subfield of Artificial Intelligence (AI). Evolution is a biological term. In biology, evolution is a change of heritable traits of a population over successive generations. Evolutionary Computing is a general term for several computational techniques that are often biologically inspired.

In the arena of AI, an Evolutionary Algorithm (EA) is a subset of EC. A typical EA normally uses mechanisms that resemble biological evolution such as *reproduction, mutation, recombination, selection, crossover* and *survival of the fittest.* These are called *genetic operators* and are defined in Appendix C. Evolutionary Algorithms are particularly suitable for search and optimization problems [34, 53]. In this work a popular form of EA called a genetic algorithm (GA) [53] is used to optimize router performance. An example of a GA is given in Appendix C.

Evolutionary Algorithms have been applied to solve MO problems with a great deal of success [26, 48, 115, 134]. The EAs to solve MO problems are called *Multi-Objective Evolutionary Algorithms* (MOEA).

An MOEA is an extension of an EA. The objective of an MOEA is to converge to the Pareto Optimal Set of the problem. To achieve this convergence, the selection process has to be able to select the individuals such that non-dominated solutions are preferred over dominated ones. Therefore, the simple roulette wheel method (defined in Appendix C) might not be appropriate. Some sophisticated methods that are specially developed for selection in MOEA have been proposed [31, 26]. Most of them use some form of Pareto ranking. The original Pareto ranking based selection method was proposed by Goldberg [53] in which the population of the EA is sorted, based on Pareto dominance, such that all non-dominated individuals are assigned the same rank. All non-dominated individuals are given the same probability for reproducing and this probability is higher than that of dominated individuals.

4.2.3.1 MOEA: Aggregating Approach

The simplest way to deal with an MO problem is to combine or aggregate multiple objectives into a single scalar function. These techniques are known as "**aggregating functions**" since they aggregate multiple objectives into a single scalar value. According to this method, an MO problem becomes

$$min \sum_{m=1}^{M} w_m f_m(\mathbf{x}) \tag{4.5}$$

where the w_m values are non-negative and represent the relative weights (importance) of the M objective functions. Usually the weights are normalized to unity, i.e., $\sum w_i = 1$ [34].

Aggregating functions methods have the advantage that they produce a single compromised solution requiring no further interaction from the decision maker [48]. Like classical approaches, this method can also get stuck in a suboptimal or dominated solution. But if the w_m values in equation (4.5) are positive for all objectives, the solution attained by this method will be Pareto optimal [94]. On the other hand, the solution obtained by this method might not be acceptable because of inappropriate settings of the weights. Making a proper choice of the weights might also be difficult. This is further discussed in section 4.3.

Alternatives to the aggregating approaches are **population based** approaches. In population based approaches (known as Vector Evaluated Genetic Algorithm (VEGA)) [55, 112], if there are M objectives, M sub-populations of size F/M (F is the population size) are selected (by means of a suitable selection process) from the whole of the old generation according to each of the objectives separately. These M sub-populations are shuffled together and then normal crossover and mutation are applied. The major drawback of VEGA is, that if there is an individual that is a good compromise solution for all objectives but not the best in any single objective, it will be discarded [48, 135]. Among the categories of MOEA, the aggregating approach is the simplest. Another advantage of this approach is its adaptability. By proper adjustments of the weights, it can adapt to diverse performance optimization objectives.

Owing to the simplicity, adaptability and the ability to converge to a Pareto optimal set of the aggregating approach, in this work it is decided to use it in dealing with the router performance optimization problem as described below.

4.3 Review of the Router Optimization Problem

This section discusses how the aggregating approach can be used to approach the router optimization problem. Although aggregating approaches are simple to understand, their implementation requires handling two major challenges. These are the scaling of the objective functions and the proper setting of the weights. They are discussed below.

It is likely that different objectives will have values of different units and orders of magnitude. For example in the car manufacturer problem in section 4.2.1 (Figure 4.6), the cost of production of a car is in dollars and the pollution can be the amount of polluted wastage material (in kilograms) it produces. Hence, direct aggregation might not be possible for these two objectives. A possible solution is to "scale" the objective functions so that they have the same range of acceptable values.

In our router optimization problem, let us consider the performance optimization objective of maximizing utilization and minimizing the packet drop rate. In both cases the range is [0, 1]. However a utilization of 0.5 might not be comparable to a packet drop rate of 0.5. This is because a packet drop rate of 0.5 is very high and might not be acceptable to many networks. In this case we can set up thresholds for acceptable values for utilization and packet drop rate. For example, let us assume the range of the packet drop rate is in [0, 0.02]. In this case a packet drop rate of 0.02 is scaled to be 1 and a packet drop rate of 0.01 is scaled to 0.5. Thus the fitness function for this simple router optimization problem becomes

$$min\,(w_1 l' + w_2(-\rho')) \tag{4.6}$$

where l' and ρ' are the scaled values of packet drop rate and utilization respectively. Similarly to l and ρ, the other objectives q and v in our router optimization problem can be scaled.

The scaling of objectives depends largely on the system requirements. A scaling that is appropriate in one system might not be appropriate to other systems. It depends on the performance objective for a particular network. In this work an AQM system is developed and the aggregating approach is used to simulate the performance of the developed scheme (Chapter 7).

The second challenge is to set up the weights. Setting up an appropriate weight vector also depends on the scaling function used for each of the objectives. There is no simple way to determine the most appropriate weight vector. But if the scaling is done properly, the complexity of setting up a proper weight vector can be reduced. In the case of the router optimization problem, the weight vector depends on the system requirements. For example, in a data network, the main optimization objective may be utilization. In a VOIP network, the most important objective function may be the drop probability. Therefore, in case of a data network, w_2 should be greater than w_1 in equation (4.6). On the other hand, in case of a VOIP network, w_1 should be greater than w_2.

4.4 Conclusion

This chapter reviewed fuzzy logic and evolutionary computing, two kinds of soft computing technique. In the first part fuzzy logic and fuzzy logic systems were reviewed. In the second part classical approaches and evolutionary approaches for solving MO problems were reviewed. The use of multi-objective evolutionary algorithms for solving the router performance optimization problem was explored. In chapter 7 an AQM system based on the fuzzy logic based traffic models that are developed in chapter 5 is proposed.

Chapter 5

Fuzzy Logic Based Traffic Models

5.1 Introduction

In this chapter two models: *a fuzzy state model* and *a fuzzy group model* of network traffic are proposed. As stated in Chapter 2, traffic models have many applications. They are used extensively in traffic control and performance optimization. Some router queue management techniques were discussed in Chapter 3.

The Fuzzy State Model models the current state of network traffic at an observation point and can be used to generate control signals for traffic sources. It is used to develop an Active Queue Management (AQM) scheme.

The Fuzzy Group Model models traffic flows of any duration. It models both single user and aggregate traffic. This model introduces a parameter named R that reflects the overall characteristics of the traffic. The group model, along with the R parameter is used in the analysis of network traffic. In order to keep these models simple, fast and secure this work restricts to data sets measured by port based passive measurements (section 3.4.1 and 3.4.3) only. For reasons of privacy, no user sensitive data such as packet payload (section 3.4.2) is used.

5.2 The Fuzzy State Model

This model represents network traffic as a fuzzy time series. This work presents a novel approach to modelling Internet traffic using fuzzy *state* information and fuzzy *trend* information. Depending on the packet arrival rate, the state of a traffic stream is described using natural linguistic terms such as "very quiet", "greedy" and "in burst". The trend of a traffic stream is also described with human understandable terms, such as "rising", "falling", "rising sharply" and "constant". In the rest of this thesis, linguistic variables and fuzzy sets are named with upper case letters. Packet arrival rates at present and in the recent past, determine the trend and the state of the traffic stream. For example, at time t, $t-1$ and $t-2$, the number of packets of a traffic stream arriving at an observation point is N_t, N_{t-1}, N_{t-2} respectively. If $N_t > N_{t-1} > N_{t-2}$, this indicates that number of packets arrived at the observation point is increasing. In this case it can be said that, the trend of the traffic stream is increasing. If $N_t < N_{t-1} < N_{t-2}$, this indicates that the number of packets arrived at the observation point is deceasing. This means the trend of the traffic stream is decreasing. Similarly, if $N_t - N_{t-1} > N_{t-1} - N_{t-2}$, it can be said the trend of the traffic stream is rising sharply.

Let the linguistic variable TREND represent the trend of a fuzzy series. In our capital letter notation, "the trend is constant" is equivalent to the fuzzy statement:

$$\mu_{CONSTANT}(TREND) = 1 \tag{5.1}$$

Equation (5.1) says, the linguistic variable TREND is a member of the fuzzy set CONSTANT with degree of membership 1. Similarly, let the linguistic variable STATE represent the state of a fuzzy time series. "The traffic is quiet" is equivalent to

$$\mu_{QUIET}(STATE) = 1 \tag{5.2}$$

Equation (5.2) means, the linguistic variable STATE is a member of fuzzy set QUIET with membership 1. The state and the trend information are represented by *state fuzzy sets* and *trend fuzzy sets* respectively. That is, the variable TREND can be a member of any number of trend fuzzy sets and the variable STATE can be a member of any number of state fuzzy sets. In equation (5.1), CONSTANT is a trend fuzzy set and in equation (5.2), QUIET is a state fuzzy set.

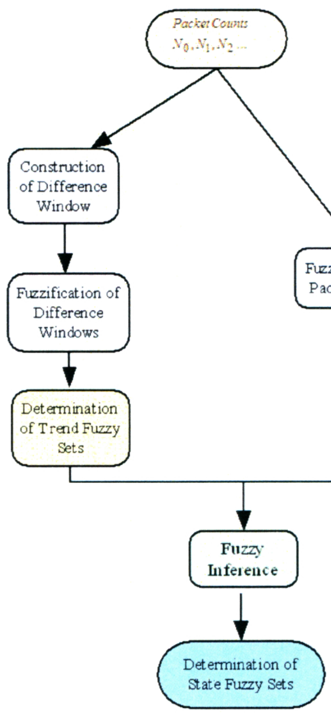

Packet Counts
$N_0, N_1, N_2 ...$
Construction of Difference Window
Fuzzification of Difference Windows
Determination of Trend Fuzzy Sets
Fuzzification of Packet Counts
Fuzzy Inference
Determination of State Fuzzy Sets

Figure 5.1: Logical Block Diagram of the Fuzzy State Model

The goal of the fuzzy state model is to compute the membership of the variable STATE for the state fuzzy sets for a given series $\{N_i\}_{i>0}$, where N_i represents the packet counts at the *ith* time interval. The input of the model is the packet arrival statistics. That is, the system only measures the packet counts. It then computes the membership values of variable TREND in corresponding trend fuzzy sets. Finally from TREND and the current packet arrival pattern the membership values of STATE in state fuzzy sets are determined.

The trend and the state information of a traffic stream can be useful in a router queue management scheme. As discussed in section 3.3, most of the active queue management schemes predict congestion based on the instantaneous or average queue size. This approach may lead to inefficient management of the buffer. For example, let us consider two cases:

- case one: the packet arrival rate is increasing and the router buffer becomes half occupied.
- case two: the packet arrival rate is decreasing and the router buffer is also half occupied.

In both cases, existing AQM schemes will use a similar drop function. However, it may be that in the second case, where the packet arrival rate is decreasing, the traffic arrival rate will decrease further and the queue length will be smaller automatically. In our fuzzy state model, however, our aim is to model the trend and the state of a traffic stream so that the queue management scheme can use the trend and the state information to determine a more appropriate packet drop probability. In a high speed network, a router has a very little time to make queuing and routing decisions. This is why our state model was developed so that it minimizes the number of complex mathematical calculations.

The block diagram of the state model is shown in Figure 5.1. The input of the model is a stream of arriving packets. At first a difference window is constructed from the arriving packet counts. This difference window and the packet counts are fuzzified to determine the trend fuzzy sets and then by means of fuzzy inferencing, the state fuzzy sets are determined. The entire process is described below.

5.2.1 Construction of Difference Windows

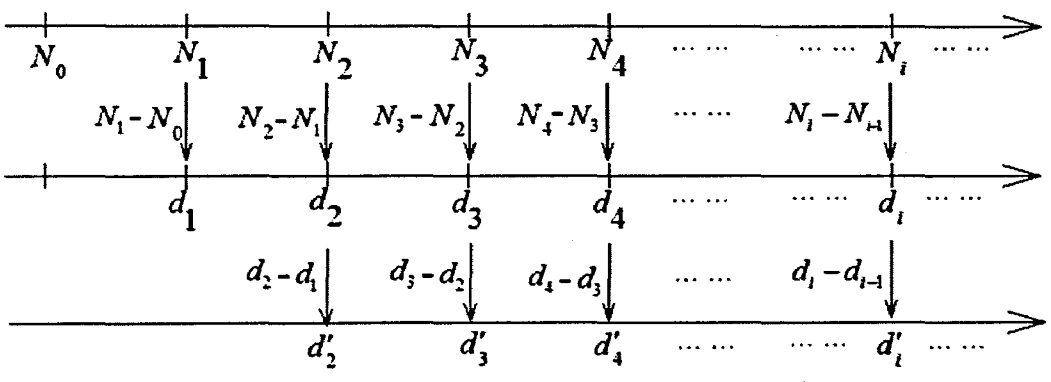


Figure 5.2: Construction of difference windows from the arriving packet counts.

Difference window values are simply the difference between two consecutive packet counts. The differences between two consecutive packet counts determine the *first order* difference windows and are denoted by variables $\{d_t\}$. The *second order* difference windows, denoted by $\{d'_t\}$, are the difference between two consecutive d_t. That is,

$$\begin{aligned} d_t &= N_t - N_{t-1} \\ d'_t &= d_t - d_{t-1} \end{aligned} \tag{5.3}$$

where N_t is the packet count at the t_{th} time bin. Figure 5.2 shows how the difference window for a given traffic trace is constructed. In the next step the N_t, d_t and d'_t are fuzzified.

5.2.2 Fuzzification of Difference Windows and Packet Counts

Fuzzification is the first step of any fuzzy logic based system (Figure 4.3). The crisp values d_i, d'_i and N_i are fuzzified using suitable membership functions (section 4.1.1). Because of its simplicity and wide acceptance, in this work a triangular membership function is used.

Using triangular membership functions, d_i and d'_i are mapped into POSITIVE, ZERO and NEGATIVE fuzzy sets. The membership function for fuzzy set ZERO is given by

$$\mu_Z(u) = \begin{cases} 0 & u \leq Z_2 \\ \frac{Z_2 - u}{Z_2} & Z_2 < u < 0 \\ \frac{Z_1 - u}{Z_1} & 0 \leq u < Z_1 \\ 0 & u \geq Z_1 \end{cases} \tag{5.4}$$

Similarly the membership function for the fuzzy set POSITIVE is given by

$$\mu_P(u) = \begin{cases} 0 & u \leq 0 \\ \frac{u}{P_1} & 0 < u < P_1 \\ 1 & u \geq P_1 \end{cases} \tag{5.5}$$

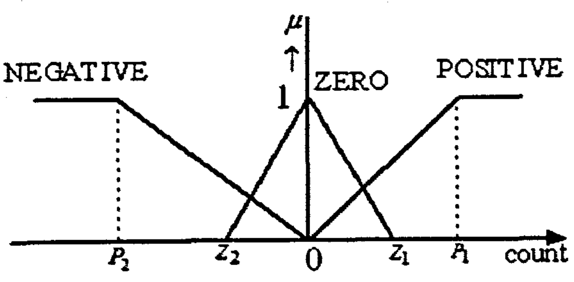


Figure 5.3: Triangular Membership Function for POSITIVE, ZERO and NEGATIVE Fuzzy Sets for d_i and d'_i.

and the membership function for NEGATIVE is given by

$$\mu_N(u) = \begin{cases} 1 & u \leq P_2 \\ \frac{u}{P_2} & P_2 < u < 0 \\ 0 & u \geq 0 \end{cases} \tag{5.6}$$

The membership functions for the ZERO, POSITIVE and the NEGATIVE fuzzy sets are shown in Figure 5.3.

Similarly to d_i and d_i', N_is are mapped into the SMALL, MEDIUM and LARGE fuzzy sets as shown in Figure 5.4. Mathematically, they are given by

$$\mu_S(u) = \begin{cases} 1 & u \leq V_1 \\ \frac{V_0-u}{V_0-V_1} & V_1 < u < V_0 \\ 0 & u \geq V_0 \end{cases} \tag{5.7}$$

$$\mu_M(u) = \begin{cases} 0 & u \leq V_2 \\ \frac{u-V_2}{V_0-V_2} & V_2 < u < V_0 \\ \frac{V_3-u}{V_3-V_0} & V_0 \leq u < V_3 \\ 0 & u \geq V_3 \end{cases} \tag{5.8}$$

$$\mu_L(u) = \begin{cases} 0 & u \leq V_0 \\ \frac{u-V_0}{V_4-V_0} & V_0 < u < V_4 \\ 1 & u \geq V_4 \end{cases} \tag{5.9}$$

By fuzzifying, each of the three (fuzzy) categories is able to represent a large range of values of d_i, d_i' or N_i. This greatly reduces the computation complexity. Once d_i, d_i' and N_i are fuzzified, the next step is to determine the membership values of TREND in corresponding trend fuzzy sets.

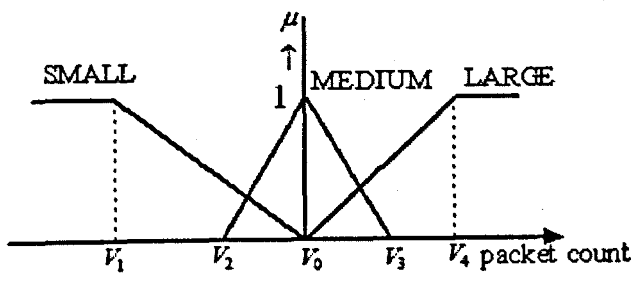


Figure 5.4: Triangular Membership Function for SMALL, MEDIUM and LARGE fuzzy sets for N_i

5.2.3 Determination of Trend Fuzzy Sets

Table 5.1: Rule-base for Fuzzy Inference Module to Determine the Trend Fuzzy Sets

No	d_{i-1}	d_i	d'_i	TREND
1	POSITIVE	POSITIVE	NEGATIVE	RISING SLOWLY
2	POSITIVE	POSITIVE	ZERO	RISING
3	POSITIVE	POSITIVE	POSITIVE	RISING SHARPLY
4	POSITIVE	NEGATIVE	NEGATIVE	CREST
5	NEGATIVE	NEGATIVE	NEGATIVE	FALLING SHARPLY
6	NEGATIVE	NEGATIVE	POSITIVE	FALLING SLOWLY
7	NEGATIVE	NEGATIVE	ZERO	FALLING
8	NEGATIVE	POSITIVE	POSITIVE	TROUGH
9	ZERO	ZERO	ZERO	CONSTANT

The trend of a traffic time series at any time instant is determined with the help of the fuzzified values of difference windows and the arriving packet counts.

Once d_i, d'_i and N_i are fuzzified, the trend fuzzy set can be determined using fuzzy inference rules. Perhaps the oldest and the most popular inferencing technique is *min* inferencing. There are other modern and sophisticated inferencing rules such as transductive knowledge based fuzzy inferencing [114], adaptive neuro-fuzzy inference system (ANFIS) [78]and Sequential Adaptive Fuzzy Inference System (SAFIS) [109]. Among all the inferencing techniques, *min* inferencing which requires computing the minimum of two membership values is the fastest. In this work *min* inferencing is used. This is because, as it is intend to use the model in router buffer management scheme, the quickest method possible should be used. The rule matrix for fuzzy inference is shown in Table 5.1. The trend of the series, expressed by the linguistic variable TREND is the output of the fuzzy inference of Table 5.1.

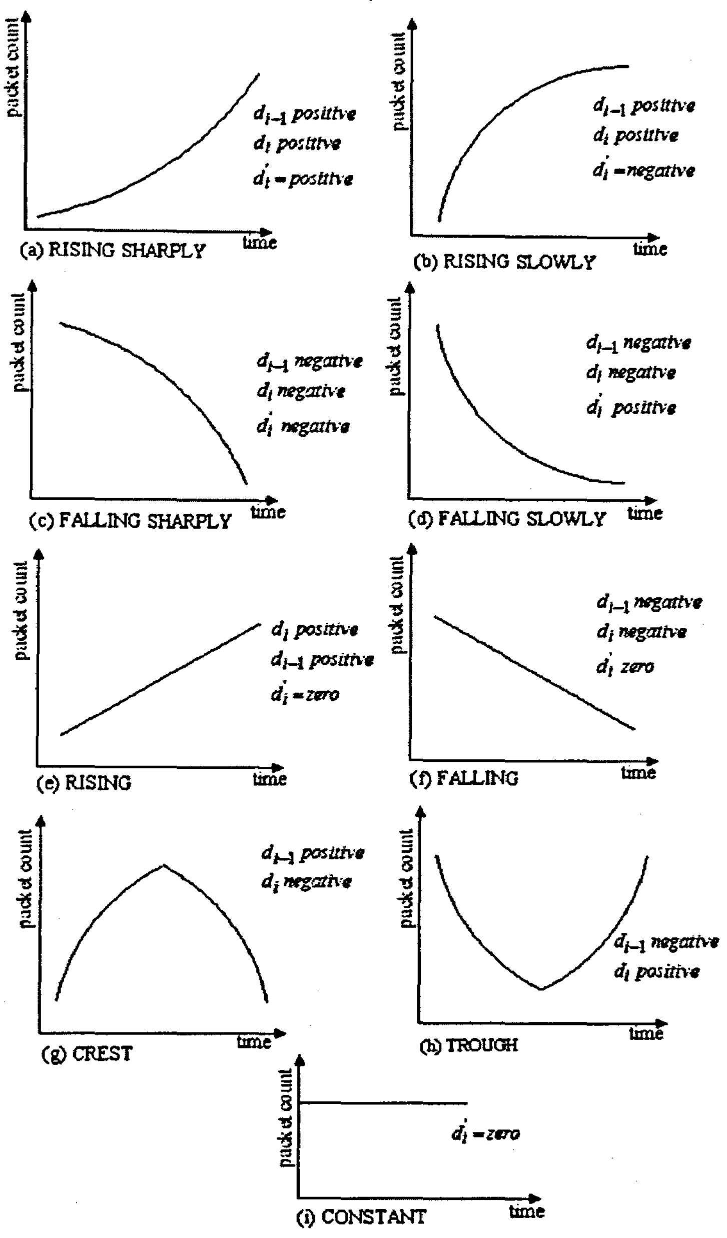


Figure 5.5: Various Trend Fuzzy Sets

Table 5.1 shows that there are 9 possible trend fuzzy sets. These are: RISING SHARPLY, RISING SLOWLY, FALLING SHARPLY, FALLING SLOWLY, RISING, FALLING, CREST, TROUGH and CONSTANT. These state are graphically shown in Figure 5.5. These are determined from the values of d_i, d_{i-1} and d'_i by observation. For example in Figure 5.5, the TREND is RISING SLOWLY, if both d_1 and d_2 are *positive* and $d'_1$is *negative*. The term "d_1 is positive" means d_1 has non-zero membership in the POSITIVE fuzzy set. Similarly other fuzzy sets are determined. If the packet arrival sequence at a time does not match any rule, the TREND is assumed to be unchanged.

To illustrate the inferencing, let us consider the following example

Let

The number of packets arriving in a sequence	=	40, 60, 70
That is $\{N_0, N_1, N_2\}$	=	$\{40, 60, 70\}$

Therefore,

$d_1 = 60 - 40 = 20$	and	$\mu_Z(d_1) = 0.2,$	$\mu_P(d_1) = 0.8$	(Fig 5.6)
$d_2 = 70 - 60 = 10$	and	$\mu_Z(d_2) = 0.6,$	$\mu_P(d_2) = 0.4$	(Fig 5.6)
$d'_2 = 10 - 20 = -10$	and	$\mu_Z(d'_2) = 0.6,$	$\mu_N(d'_2) = 0.4$	(Fig 5.6)

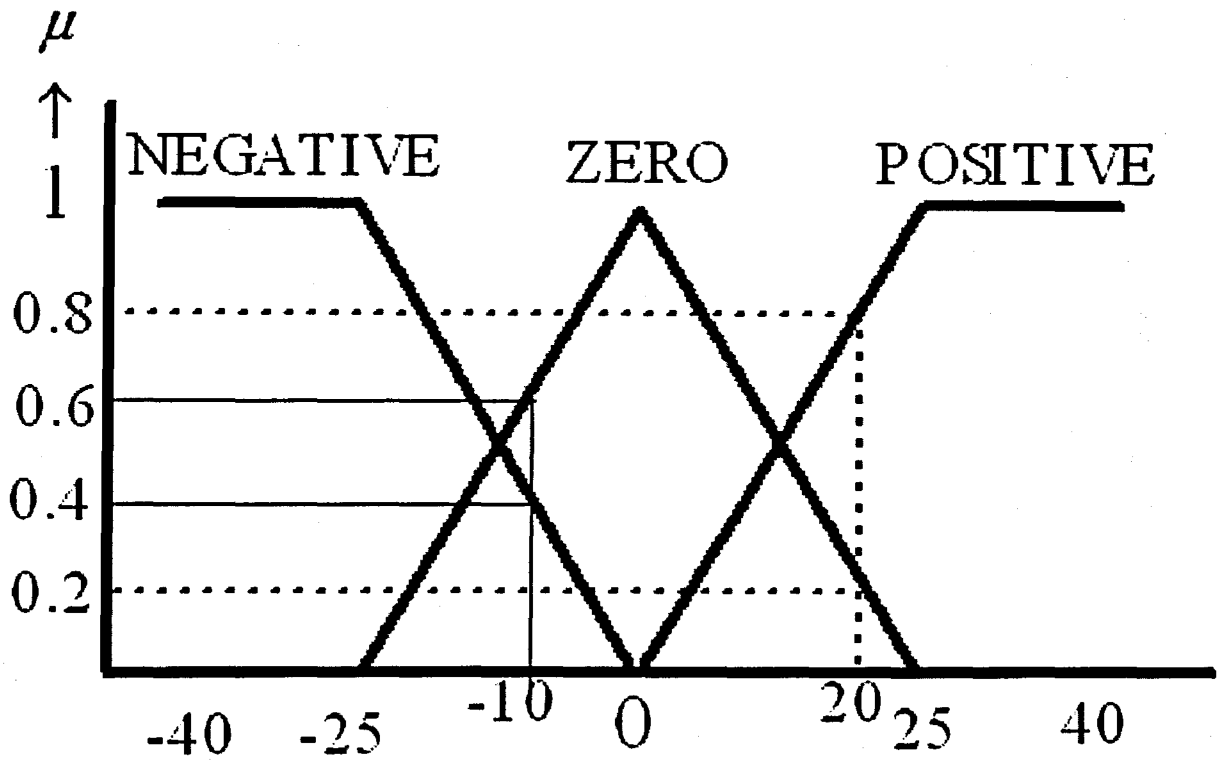


Figure 5.6: Membership functions and grade of membership for the fuzzification of {40,60,70} traffic stream

The membership values of *TREND* in trend fuzzy sets are computed from Table 5.1. In this example, d_1 and d_2 have non-zero memberships in *ZERO* and *POSITIVE* fuzzy sets. Similarly, d'_2 has non-zero memberships in *ZERO* and *NEGATIVE* fuzzy sets. Therefore, there can be eight different combinations of

(d_1, d_2, d_2'). Among these eight combinations the only combinations to fire rules in table 5.1 are:

$(ZERO,\ ZERO,\ ZERO)$;

$(POSITIVE,\ POSITIVE,\ ZERO)$; and

$(POSITIVE,\ POSITIVE,\ NEGATIVE)$.

These combinations match rules 9, 2 and 1 respectively.

Hence $TREND$ is,

$$\begin{aligned}
\text{CONSTANT with membership (rule9)} &= min(\mu_Z(d_1), \mu_Z(d_2), \mu_Z(d_2')) \\
&= min(0.2, 0.6, 0.6) = 0.2 \\
\text{RISING with membership (rule2)} &= min(\mu_P(d_1), \mu_P(d_2), \mu_Z(d_2')) \\
&= min(0.8, 0.4, 0.6) = 0.4 \\
\text{RISING SLOWLY with membership (rule1)} &= min(\mu_P(d_1), \mu_P(d_2), \mu_N(d_2')) \\
&= min(0.8, 0.4, 0.4) = 0.4
\end{aligned}$$

In the above example, the variable $TREND$ of the traffic stream {40,60,70} is: CONSTANT with membership 0.2, RISING with membership 0.4 and RISING SLOWLY with membership 0.4.

Once the trend fuzzy sets are determined, the next step is to find the state of the traffic stream.

5.2.4 Determination of State Fuzzy Sets

The membership values of variable TREND in trend fuzzy sets and the current packet count are used to determine the state fuzzy sets. At first a compound label is formulated. The compound label consists of current trend information and current packet arrival information. The compound label is a pair of membership values of two terms. The first term is the value of $TREND$ with corresponding grade of membership. The second is the fuzzy set that represents the packet counts with corresponding grade of membership. One example of such a label is {0.6 RISING SHARPLY, 0.9 LARGE}. This label indicates that the TREND is RISING SHARPLY with membership 0.6 and the number of packets which has arrived is LARGE with membership 0.9. The label can be viewed as an "event".

When such an event acts on a traffic stream, the state of the traffic is transformed into a new state.

The state of the traffic is represented by the linguistic variable STATE. The variable STATE reflects the activity level of the traffic. STATE can be a member of more than one state fuzzy set. The state fuzzy sets have easily understandable linguistic terms, such as QUIET or BURST.

The compound label acts on a traffic stream and transforms it into new state. For example, the compound label {RISING SHARPLY, LARGE} acting on QUIET, transforms it into BURST. Denoting the acting operator by symbol $\uplus$, this rule becomes,

$$\{\text{RISING SHARPLY, LARGE}\} \uplus \text{QUIET} \Rightarrow \text{BURST} \tag{5.10}$$

There can be many rules similar to equation (5.10). The rules may vary from system to system. The system designer can define any number of such rules based on the system requirement. One major aim of this research is to use the model for traffic analysis and router performance optimization. The states and rulebase used in this work are given in equation (5.11). In (5.11), the symbol X indicates "DON'T CARE" or any value and X_B indicates any value other than BURST. This work calls the rules in (5.11) the *state transition rules*.

$$\begin{array}{lll}
\text{R1: } \{\text{RISING SHARPLY, LARGE}\} \uplus X_B & \Rightarrow & \text{BURST} \\
\text{R2: } \{\text{RISING SHARPLY, LARGE}\} \uplus \text{BURST} & \Rightarrow & \text{GREEDY} \\
\text{R3: } \{\text{FALLING SHARPLY, MEDIUM}\} \uplus X & \Rightarrow & \text{DOWNWARD ALARM} \\
\text{R4: } \{\text{FALLING SHARPLY, SMALL}\} \uplus X & \Rightarrow & \text{DOWNWARD QUIET} \\
\text{R5: } \{\text{CONSTANT, SMALL}\} \uplus \text{DOWNWARD QUIET} & \Rightarrow & \text{QUIET} \\
\text{R6: } \{\text{RISING SHARPLY, MEDIUM}\} \uplus \text{QUIET} & \Rightarrow & \text{UPWARD ALARM} \\
\text{R7: } \{X\text{, MEDIUM}\} \uplus \text{UPWARD ALARM} & \Rightarrow & \text{ALARM}
\end{array} \tag{5.11}$$

Equation(5.11) shows that there are seven different states fuzzy sets that are used in this work. These are: BURST, GREEDY, DOWNWARD ALARM, DOWNWARD QUIET, QUIET, UPWARD ALARM and ALARM. When traffic is in the QUIET state, this indicates that the traffic has low activity, or equivalently, the TREND is constant and N_i is SMALL. Similarly, if suddenly a large number of packets arrives in a relatively short amount of time, the traffic stream is in the BURST state. The ALARM state indicates that traffic activity is increasing alarmingly. The state DOWNWARD QUIET indicates that the packet arrival rate is decreasing and the current packet count is small. In this case an AQM scheme might think of stopping packet dropping. The state DOWNWARD ALARM indicates that the packet arrival rate is decreasing and still the current packet count is not small enough to termed as QUIET.

Fuzzy inference rules are used to compute the membership of the variable STATE to each of the state fuzzy sets. For example, using *min* inference, according to rule (5.10), if the label {0.6 RISING SHARPLY, 0.9 LARGE} acts on a QUIET state with membership 0.7, it will transform the STATE into BURST with membership $= min(0.6, 0.9, 0.7) = 0.6$. That is,

$$\{0.6 \text{ RISING SHARPLY}, 0.9 \text{ LARGE}\} \uplus \{0.7\,\text{QUIET}\} \Rightarrow 0.6\,\text{BURST} \tag{5.12}$$

The state transition diagram associated with the rules in (5.11) is shown in Figure 5.7.

Let us consider the state transition rule

$$\{\text{RISING SHARPLY, LARGE}\} \uplus \text{BURST} \Rightarrow \text{GREEDY} \tag{5.13}$$

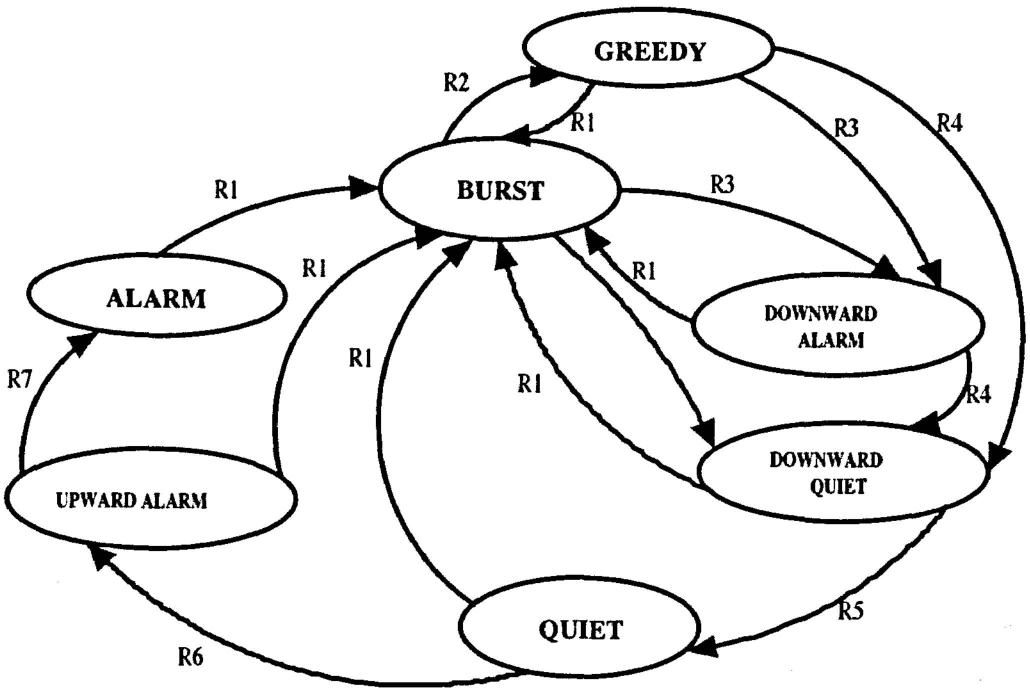


Figure 5.7: Various State Fuzzy States and Their Transitions

This states that the traffic is already in BURST and within this state the trend is RISING SHARPLY and the number of packets which has arrived is large. This reflects the burst-within-burst which is a common phenomenon in self-similar traffic (section 2.10). Burst-within-burst is not observed in a Poisson traffic. That is for Poisson traffic, this rule is unlikely to be matched.

In equation (5.11), the third rule states that, if the trend is FALLING SHARPLY and the number of packets which has arrived is MEDIUM, the state of the traffic stream is changed to DOWNWARD ALARM. This state represents the situation when the packet arrival rate in a traffic stream is decreasing but still not low enough to be considered as QUIET. If the trend is FALLING SHARPLY and the number of packets which has arrived is SMALL, the traffic state is DOWNWARD QUIET. The state of a traffic stream modelled in this way can be used by a router in its queue management scheme. For example, if the state of an incoming traffic stream is DOWNWARD ALARM, the router may lower its packet drop probability or if the state is DOWNWARD QUIET, the router may not need to drop any packets.

5.2.5 Discussion of the State Model

One of the main characteristics of the state model is its simplicity. As it is designed to be used in the router optimization scheme, it has to be simple and quick so that it can be implemented in the framework of a router without significantly increasing the router's load. The state model does not perform any complex computation. Both fuzzification and inferencing use simple arithmetic. The use of *min* inferencing also helps it to be simple.

The state model uses only the packet counts. It does not make any active or content-aware measurements. It does not even capture packet headers. Thus it is simple, fast and secure.

The traffic state can be an important parameter for a router Queue Manager. For example, if the traffic state is QUIET, the Queue Manager does not need to drop any packets. In Chapter 7 the application of this fuzzy state model to assist Queue Managers to determine an appropriate packet drop probability is discussed.

5.3 The Fuzzy Group Model

No suitable mathematical formula or quantity that can be used as a measure of the burstiness of a traffic stream exists. Section 2.10 discusses some of the then available approaches for measuring burstiness. Although the energy plot (section 2.10.1) can be a useful measure of burstiness, it produces a graphical presentation only, not a mathematically derived parameter of the traffic stream. One of the aims of this research is to develop a fuzzy logic based traffic model which can provide a mathematical quantity that can be used to characterize traffic streams and can also be used as a measure of burstiness and an indicator of queuing performance. In this work this model is named as *fuzzy group model.*

A packet is the smallest building block or unit of a traffic trace. In a TCP/IP communication, headers are added to the user data by means of encapsulation. In a TCP/IP network, a packet can be identified by the following six attributes:

1. source port (added in the transport layer);

2. destination port (added in the network layer);

3. transport protocol used (added in the transport layer);
4. source IP address (added in the network layer);
5. destination IP address (added in the network layer); and
6. time-stamp (added in the datalink layer).

That is, a packet is identified by six attributes: time-stamp (t), source address (sa), source port (sp), transport protocol ($proto$), destination address (da) and destination port (dp).

$$p = (t, sa, sp, proto, da, dp) \tag{5.14}$$

In order to be a valid packet all six attributes must be present.

Suppose a source sends a number of packets to a destination. After some idle time, the source sends more packets to the same destination. The second set of packets is considered to be in the same flow as the first if the idle time between them is smaller than a threshold τ_f. If the idle time is greater than or equal to τ_f, flow time-out occurs and the two sets of packets are considered to be in different flows. The threshold τ_f is known as the flow time-out threshold.

A flow is a sequence of packets in a traffic trace that have the same source and destination addresses and ports, same transport protocol and the difference in time-stamp of any two consecutive packets is less than τ_f.

$$f = \{p_i \,|\, p_i(sa, sp, proto, da, dp) = p_{i-1}(sa, sp, proto, da, dp)\ ,\ t_i - t_{i-1} < \tau_f\} \tag{5.15}$$

where $\{\}$ denotes a sequence of packets, p_i is the ith packet and τ_f is the flow time-out threshold expressed in time-unit.

A value of τ_f can be anything over 1 minute [11, 41, 67]. The authors in [11, 22, 41] studied flow time-out values of 5 minutes, 15 minutes and 20 minutes and found little difference in. Jain and Routhier in [67] used a value of 500 milliseconds. In this work, the value of τ_f is assumed to be 10 minutes.

In real traffic, packets may come in batches. The fuzzy group model treats a batch as a *group*. A *packet group* is the largest possible sequence of packets in a stream so that the time interval between any two consecutive packets within the sequence is smaller than a threshold. This work has defined two types of groups: f-group and T-group. These groups are shown graphically in Figure 5.8 and are defined below.

f-group

When consecutive packets in the same flow form a group it is called in an f-group. Suppose the sequence of packets $\{p_k \mid k = 1, 2, \cdots\}$ forms a flow f_n. The jth f-group, g_j, in f_n is defined as

$$g_j = \{p_i \in f_n \mid (t_i - t_{i-1}) < \tau_g\} \tag{5.16}$$

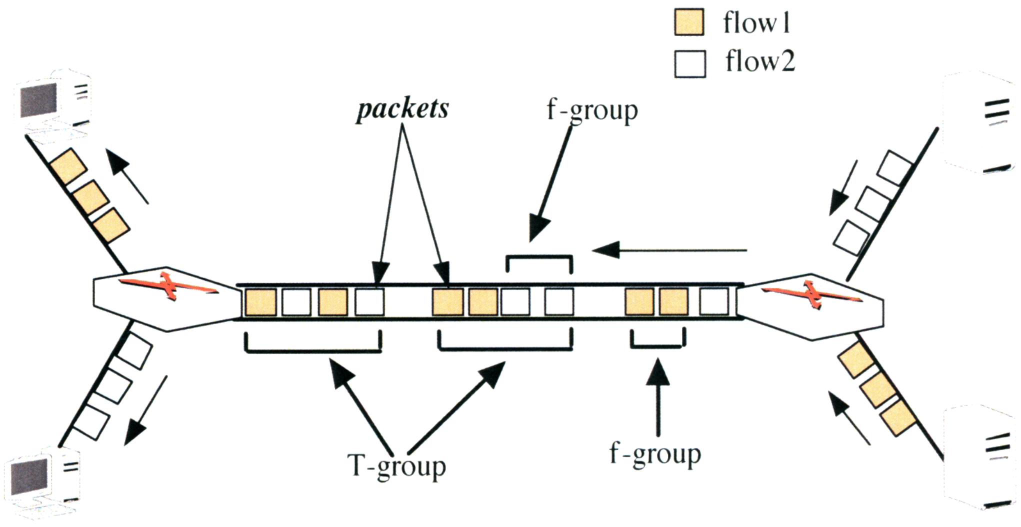


Figure 5.8: Packets, T-groups and f-groups.

where τ_g is the *f-group time-out threshold*. This indicates that if the difference in time-stamps of two consecutive packets in a flow is greater than or equal to τ_g, the packets belong to different f-groups.

T-group

When consecutive packets in the aggregate traffic stream form a group, it is called in a T-group. Suppose the sequence of packets $\{p_l \mid l = 1, 2, \cdots\}$ forms a traffic stream Γ. A T-group G_j in Γ is defined as

$$G_j = \{p_i \in \Gamma \mid (t_i - t_{i-1}) < \tau_G\} \tag{5.17}$$

where τ_G is the *T-group time-out threshold* and is expressed in time unit. This indicates that if the difference in time-stamps of two consecutive packets in a trace is greater than or equal to τ_G, the packets belong to different T-groups.

The nature and the characteristics of traffic or even a flow change dynamically. A flow consists of packets and the distribution of these packets determine the characteristics of the flow. Due to the inherent mechanism of TCP, the packet transmission rate can vary largely over time. Our aim is to "capture" the instantaneous dynamics of the flow/traffic with the help of the group model. The traffic is modelled as a sequence of T-groups and a flow as a sequence of f-groups. The distribution of these groups determines the characteristics of the traffic and the performance of the traffic on a router. For example, let us consider a traffic trace consisting of a large number of T-groups passing through a router. If the time spacing between these groups is smaller than the time required by the router to serve the packets in a group, there will be packet loss or queuing delay in the router. On the other hand, traffic passing through the router with small groups with sufficient and uniform spacing will be processed smoothly by the router, resulting in little or no queuing. These distributions of the groups are modelled using fuzzy logic and then a *performance index* that reflects the performance of traffic on a router is introduced.

5.3.1 Modelling a Flow

A single user traffic flow is modelled as a sequence of f-groups. That is,

$$f = \{g_i\}_{i>0} \tag{5.18}$$

where g_i is the ith f-group within the trace.

Each g_i is characterized by 3 parameters, as:

$$g_i = < t_i, b_i, d_i > \tag{5.19}$$

where t_i is the time-stamp of the first packet of the *ith* f-group. b_i and d_i are the byte count and the duration of the *ith* f-group. The duration of the f-group is the time required by the group to "pass" an observation point. This is the difference between the time-stamps of the last packet and the first packet of the f-group.

5.3.2 Modelling Aggregate Traffic

A traffic trace is modelled as a sequence of T-groups. That is,

$$\Gamma = \{G_i\}_{i>0} \tag{5.20}$$

where G_i is the *ith* T-group within the trace. In our model, the value of τ_G (T-group time-out threshold) is chosen so that it is smaller than or equal to the average time required by a router to serve one packet. That is, if φ is the average time required by the router to serve one packet, then τ_G holds the condition

$$\tau_G \leq \varphi \tag{5.21}$$

Each G_i is characterized by 3 parameters, as:

$$G_i = < T_i,\ B_i,\ D_i > \tag{5.22}$$

where T_i is the time-stamp of the first packet of the *ith* T-group at an observation point (such as the router). B_i and D_i are the byte counts and the duration of the *ith* T-group. The duration, D_i, is the difference in time-stamps of the last packet and the first packet of the T-group.

When a T-group goes through a router it can undergo one or more of the following transformations which have been named:

- NULL Transformation. The group undergoes no packet loss and no queuing. That is, there is no change in B_i and D_i.
- L Transformation. One or more packet(s) is(are) lost due to congestion. B_i is changed. D_i and T_i are also changed.
- Q Transformation. T_i is changed due to queuing. B_i is not changed, D_i may or may not change.

- QM transformation. The group is merged with another group due to queuing. To understand the QM transformation, let us consider a T-group entering into a router queue, where another T-group is already queued. It can happen that the router dispatches the two T-groups "back-to-back", making the two T-groups merge together to form one large T-group. In the case of a QM transformation, all model parameters are likely to change.

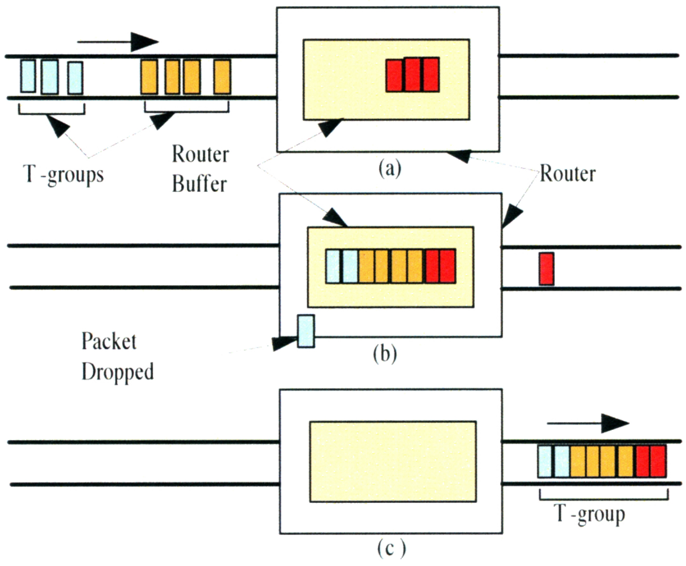


Figure 5.9: Various Transformations in a Router. (a) Two T-groups arrive at a router. (b) A packet is dropped and other packets are queued (c) A new T-group is formed.

Figure 5.9 shows how a new T-group can be formed by undergoing such transformations.

When an f-group goes through a router, one or more of the above transformations may also take place. For an f-group, in the above transformations, T_i, B_i and D_i will be t_i, b_i and d_i respectively.

These transformations in T-groups play a major role in the performance of the network. The L transformation is responsible mainly for drop-outs and retransmission. On the other hand, Q and QM transformations are responsible mainly for jitter and extreme jitter.

New parameters can be introduced to represent the measure of probability of a group undergoing these transformations. In a router with an infinite buffer, the dominant type of transformations related to performance are Q and QM transformations. A **new parameter,** R, is introduced to represent the measure of likelihood of undergoing a Q (and/or QM) transformation in such a system. The higher the value of R, the higher the probability that traffic will undergo queuing delay.

5.3.3 Determination of R

Let the inter-arrival time between the $(i-1)th$ and ith T-groups be denoted as δ_i. According to the group model, a traffic trace is a sequence of T-groups (equation (5.20)). The distribution of δ_is will affect the router's overall performance. To illustrate, suppose that at time T_1, a T-group of size B_1 bytes arrives at a router. Let s be the service rate of the router in bytes per second. That is, the router requires $1/s$ seconds to serve 1 byte of data. The router will require B_1/s seconds to serve the T-group. Let the next group arrive at the router at time T_2 ($\delta_2 = T_2 - T_1$). Therefore, if $\delta_2 < B_1/s$, it will cause the size of the queue of the router to increase, simply because the second T-group arrives before the first T-group has been served. Thus, the distribution of the δ_is and B_i, along with the router service rate s will determine the overall router performance. Let us consider the following three special cases:

- case 1: $\delta_i \gg B_{i-1}/s$
- case 2: $\delta_i \ll B_{i-1}/s$
- case 3: $\delta_i \approx B_{i-1}/s$

In case 1, the router has sufficient time to serve the $(i-1)th$ T-group before the ith T-group arrives. Hence, there will be little or no queuing in the router, but the system achieves low throughput.

In case 2, a T-group arrives before the previous T-group has been dispatched causing the router queue to increase. The overall system throughput is high in this case.

In case 3, the router operates almost at its optimum performance. The queue should be moderately small and the throughput should be high.

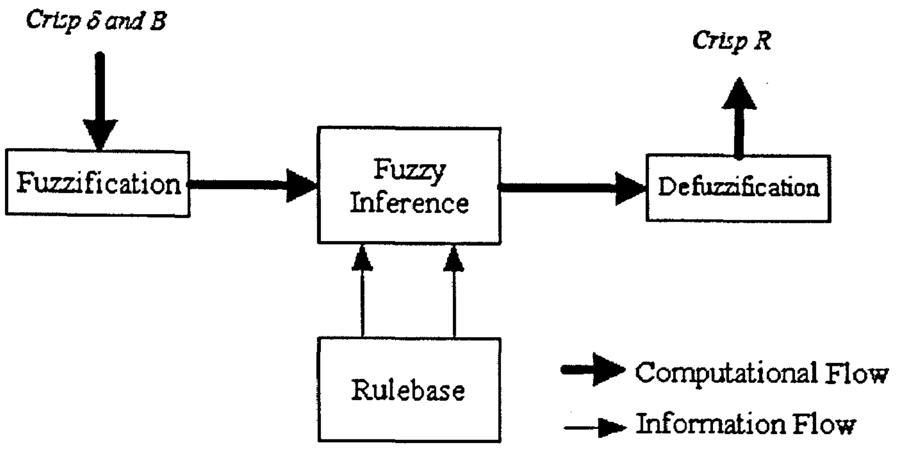


Figure 5.10: Fuzzy Logic System to Determine R

We want the R parameter to reflect the overall traffic characteristics of traffic from a performance point of view. In order to make it easily comparable to the Hurst parameter, the value of R is restricted to be between 0 and 1. Therefore, ideally R should be defined to have a value close to 0 in case 1 mentioned above where the router is likely to do the least amount of queuing. Similarly, in case 2 where the router is likely to have a large queue, R should have a value close to 1 and in case 3, R should have an interim value between 0 and 1 (preferably around 0.5). In this way, looking at the value of R, we can quickly gain an idea of the characteristics of a traffic trace from a performance point of view. Smaller values of R indicate that if the traffic is fed to a router, there will be a little or no queuing. Similarly, higher values of R indicate that there may be large queuing of packets. If traffic with a high value of R is fed to a router with finite buffer, it is likely that there will be significant amount of queuing and packet drops.

To determine R, a fuzzy logic system similar to the one described in section 4.1.2 is used. The general fuzzy logic system of Figure 4.3 is redrawn (Figure 5.10) to show the steps of determination of R.

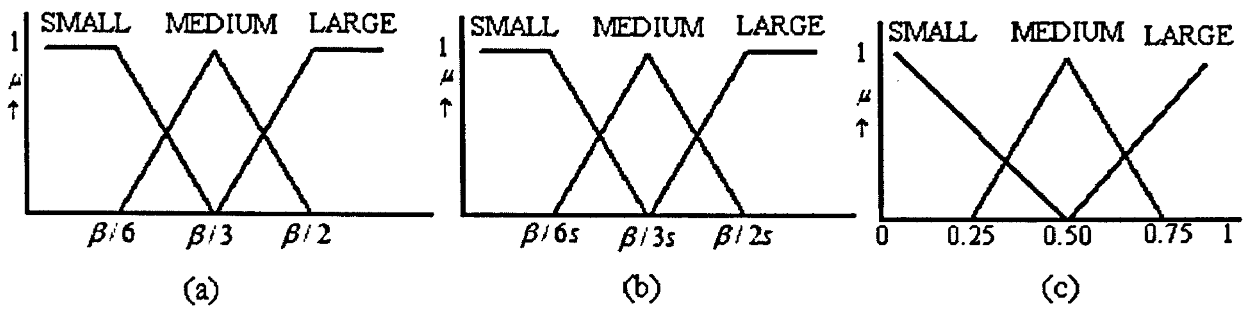


Figure 5.11: Membership Function of (a) B and (b) δ for aggregate traffic.

The first step is the fuzzification of δ and B. They are fuzzified with a triangular membership function. Each B_i and δ_i can be a member of fuzzy sets SMALL, MEDIUM and LARGE. Setting up the support (section 4.1.1), core and the universe of discourse of the fuzzy sets depends on the system. For example, a value of B which can be large for a 1 Mbps link can be too small to be considered large in a 100Mbps link. One possible approach to determine the support and universe of discourse of the fuzzy sets is to relate B and δ to the buffer capacity β.

Several of the AQM schemes (section 3.3.2) such as SRED [101] aim to keep the router queue to less than $\frac{\beta}{6}$ [101]. This value can be used as a reference point. That is, if B is smaller than $\frac{\beta}{6}$, it can be considered as fuzzy "SMALL". Using this approach, one possible triangular membership function for SMALL, MEDIUM and LARGE fuzzy sets for B is shown in Figure 5.11. Mathematically,

$$\mu_S(B) = \begin{cases} 1 & B \leq \frac{\beta}{6} \\ \frac{\frac{\beta}{3}-B}{\frac{\beta}{3}-\frac{\beta}{6}} & \frac{\beta}{6} < B < \frac{\beta}{3} \\ 0 & B \geq \frac{\beta}{3} \end{cases} \tag{5.23}$$

$$\mu_M(B) = \begin{cases} 0 & B \leq \frac{\beta}{6} \\ \frac{B-\frac{\beta}{6}}{\frac{\beta}{3}-\frac{\beta}{6}} & \frac{\beta}{6} < B \leq \frac{\beta}{3} \\ \frac{\frac{\beta}{2}-B}{\frac{\beta}{2}-\frac{\beta}{3}} & \frac{\beta}{3} < B < \frac{\beta}{2} \\ 0 & B \geq \frac{\beta}{2} \end{cases} \tag{5.24}$$

$$\mu_L(B) = \begin{cases} 0 & B \leq \frac{\beta}{3} \\ \frac{B-\frac{\beta}{3}}{\frac{\beta}{2}-\frac{\beta}{3}} & \frac{\beta}{3} < B < \frac{\beta}{2} \\ 1 & B \geq \frac{\beta}{2} \end{cases} \tag{5.25}$$

Similarly to B, the SMALL, MEDIUM and LARGE fuzzy sets for δ can be determined in relation to the router service rate s and the buffer capacity β. The router service rate is the number of bytes the router can serve in one second. A router with service rate s requires $\frac{\beta}{6s}$ seconds to serve a T-group of size $\frac{\beta}{6}$. Therefore, if δ is smaller than $\frac{\beta}{6s}$, it can be considered as fuzzy "SMALL". Using a triangular membership function, mathematically the SMALL, MEDIUM and LARGE fuzzy sets for δ are defined as

$$\mu_S(\delta) = \begin{cases} 1 & \delta \leq \frac{\beta}{6s} \\ \frac{\frac{\beta}{3s}-\delta}{\frac{\beta}{3s}-\frac{\beta}{6s}} & \frac{\beta}{6s} < \delta < \frac{\beta}{3s} \\ 0 & \delta \geq \frac{\beta}{3s} \end{cases} \tag{5.26}$$

$$\mu_M(\delta) = \begin{cases} 0 & \delta \leq \frac{\beta}{6s} \\ \frac{\delta-\frac{\beta}{6s}}{\frac{\beta}{3s}-\frac{\beta}{6s}} & \frac{\beta}{6s} < \delta \leq \frac{\beta}{3s} \\ \frac{\frac{\beta}{2s}-\delta}{\frac{\beta}{2s}-\frac{\beta}{3s}} & \frac{\beta}{3s} < \delta < \frac{\beta}{2s} \\ 0 & \delta \geq \frac{\beta}{2s} \end{cases} \tag{5.27}$$

$$\mu_L(\delta) = \begin{cases} 0 & \delta \leq \frac{\beta}{3s} \\ \frac{\delta-\frac{\beta}{3s}}{\frac{\beta}{2s}-\frac{\beta}{3s}} & \frac{\beta}{3s} < \delta < \frac{\beta}{2s} \\ 1 & \delta \geq \frac{\beta}{2s} \end{cases} \tag{5.28}$$

Table 5.2: Rule-base for Fuzzy Inferencing to Compute Output Fuzzy Sets for R

B	δ	R
LARGE	SMALL	LARGE
LARGE	MEDIUM	MEDIUM
LARGE	LARGE	SMALL
MEDIUM	SMALL	MEDIUM
MEDIUM	MEDIUM	SMALL
MEDIUM	LARGE	SMALL
SMALL	SMALL	SMALL
MEDIUM	LARGE	SMALL

Once B and δ are fuzzified, we require a set of $IF - THEN$ rules for fuzzy inference. Like the state model, *min* inference is used. The rule-base used in this work to determine R is shown in Table 5.2. Most of the rules are self explanatory. For example, the first rule states that, if B is LARGE and δ is SMALL, R is LARGE. This is the case when the router does not get enough time to serve the incoming group. The last two rules state that, if B is SMALL or δ is LARGE, R is small. This is because in both cases a router has enough time to serve the incoming group.

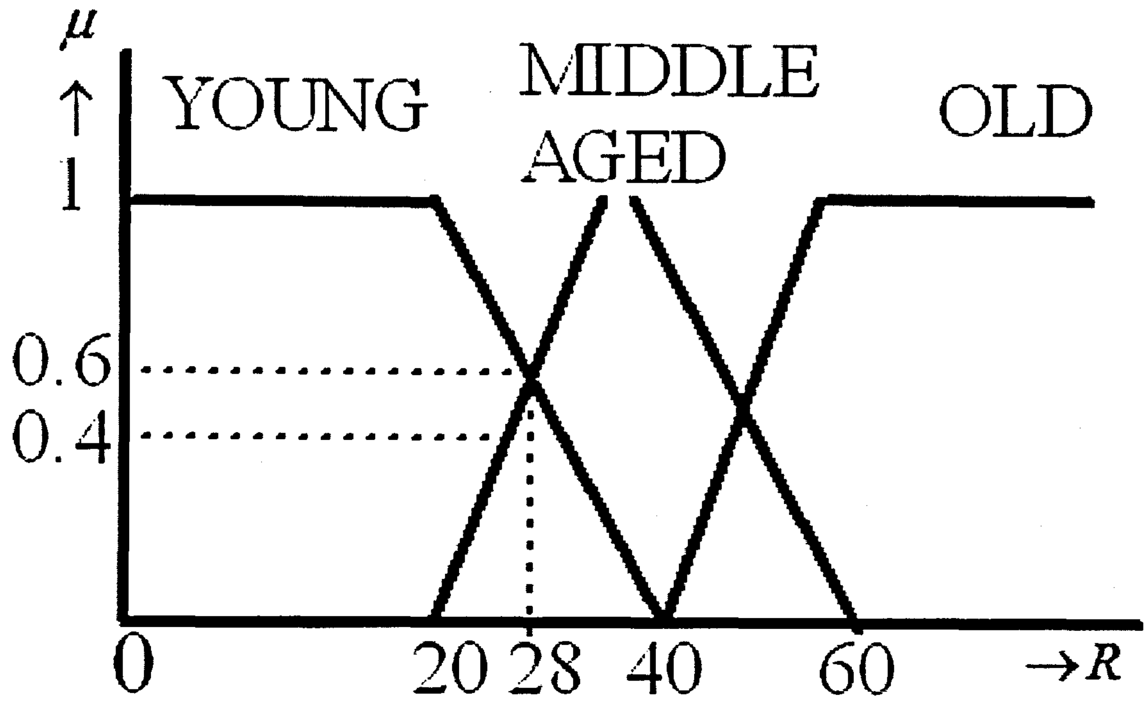


Figure 5.12: Triangular Membership Function of Output Fuzzy Sets for R: SMALL, MEDIUM and LARGE.

The final step is the defuzzification to output fuzzy sets for R to produce a crisp R. The inference module produces the output fuzzy sets SMALL, MEDIUM and LARGE for R along with the membership values. If R is allowed to have values between 0 and 1, one possible triangular membership function for R is shown in Figure 5.12. R is then calculated using centroid defuzzification (equation (4.3)). Appendix E has the pseudocode to compute R.

5.3.4 Discussion of the Group Model

In section 2.9, the general limitations of statistical and mathematical traffic models are discussed. The first reason that motivated us to develop a fuzzy logic based traffic model is that statistical models make some assumptions. The group model does not make any such statistical assumptions. A traffic trace originating from any type of network or traffic originating from any level of aggregation can be modelled by the group model. This is the advantage of using a measured approach.

The group model does not perform any active or content-aware measurement. It uses packet counts and byte counts only. Hence it is fast and secure.

The group model does not perform any complex mathematical computation. It uses a singleton fuzzifier, min inference and a centroid defuzzifier, all of which are are relatively simple and quick to compute.

Unlike the estimation of H, the estimation of R does not require any data at large scales. Hence estimation or R does not require a huge data sets.

The determination of R involves setting up the core, support and universe of discourse of SMALL, MEDIUM and LARGE fuzzy sets for the fuzzification of B, δ, and R. The values for these parameters that are used to determine R for traffic analysis are described in section 6.2.2.

5.4 Conclusion

This chapter demonstrated how a fuzzy logic based formulation can be used to model network traffic. One of the developed models introduced the R parameter. Traffic analysis using the R parameter is discussed in Chapter 6. The R parameter is defined in a way that it is related to the queuing performance. In Chapter 6 queuing performance of 84 real traces and the use of R as an indicator of queuing performance are studied.

One of the aims of this research is to develop a mathematical quantity that can be a measure of burstiness of a traffic stream. Chapter 6 also studies the burstiness of traffic traces with different R values.

A router buffer queue management scheme based on the state model is proposed in Chapter 7.

Chapter 6

Traffic Analysis Using Our Proposed Traffic Model

6.1 Introduction

Internet traffic exhibits high variability across a wide range of scales (Section 2.5.3). If we record packet counts or byte counts at an observation point on a network link on a short time scale (10 milli-seconds or less), we observe a highly variable process where packet counts or byte counts are constantly changing. It might be expected that if we increase the time scale to the seconds range, the counts would be smoother. But instead of observing a smoother pattern, we always observe that the counts are almost as variable as the finer time scales. Qualitatively, this property is termed self-similarity.

The observation of the existence of Self-Similarity (SS) and Long-Range Dependency (LRD) in both LAN and WAN traffic was a landmark which made Poisson-based computer network traffic models obsolete (section 2.5.4). The significance of this discovery is that SS and LRD have a large impact on queuing performances and traffic prediction. For example, self-similar traffic requires a larger router buffer memory than is needed for traffic following a Poisson arrival rate [119]. Higher values of H, the Hurst parameter indicate higher degrees of self-similarity. Therefore, a traffic trace with a higher H value should require a larger buffer than traces with smaller H values. In this chapter the queuing performance of several real traces with different H and R values are studied.

The existence of self-similarity indicates that Internet traffic is bursty (section 2.5.3). This burstiness can be characterized by wavelet based MRA (Multi Resolution Analysis) Energy Plots developed by Abry and Veitch (section 2.10). Although there is no precise mathematical definition of the term "burstiness", there are some statistical techniques to measure the burstiness (some were mentioned in section 2.10). Because of their simplicity and accuracy, MRA energy plots are used to measure the burstiness of the traffic in the work described below.

In this chapter the fuzzy group model (and the R parameter) that was developed in Chapter 5 is used to perform traffic analysis. The R parameter is defined in a way that it can be a measure of burstiness and of the queuing performance of a traffic stream. In this chapter the queuing performance and burstiness analysis of real traffic traces with different H and R values are studied. Specifically the following are examined:

1. **Experiment 1**: A comprehensive analysis of 84 real computer network traffic traces available from the world wide web collected at different time of the day and days of the week. Among these 84 traces, 28 traces are from 2001, 28 from 2003 and 28 from 2005. The purpose of this analysis is to visualize the H and R values of traffic traces of different years (section 6.2). This study gives an overview of how H and R values are changing with the evolution of the network. H and R values are studied as a function of link utilization for the 84 traces. This study also examines the relationship of H and R values with the total number of active connections and examines the stationarity of the traces.

2. **Experiment 2**: Trace driven queuing analysis of nine out of 84 real traces (three each from 2001, 2003 and 2005). Queuing performance graphs of several other AMP 2005 traces are shown in Appendix D. The queuing performance of two generated self-similar traffic traces are also studied. The traces are generated using the method of section 2.5.2.2. The queuing performances of traces with different H and R values are studied.

3. **Experiment 3:** MRA energy plots of six real traces. The traces from AMP 2005 are used in this experiment. Energy plots are used as a measure of burstiness (section 2.10.1). The purpose of this study is to find any relationship between the burstiness of a trace with H or R.

The next three sections describe these experiments.

Table 6.1: Traces Used for the Analysis.

Name	Year	Source URL	Access Date
Auckland IV	2001	http://pma.nlanr.net/Traces/long/auck4.html	04-01-2006
Auckland VIII	2003	http://pma.nlanr.net/Traces/Traces/long/auck/8/	04-01-2006
AMPI	2005	http://pma.nlanr.net/Traces/Traces/long/apth/1/	22-04-2006

6.2 Experiment 1: *H* and *R* Parameters for Real Traces

In this experiment the H and R values of publicly available traffic traces are computed. But publicly available traces with empirical data are now becoming more and more limited. This is mainly because network link speeds have increased dramatically in recent years from 10 Mbps to 1 Gbps or even higher. Capturing even just the packet headers from such high speed links requires a huge amount of processing power and disk storage. Another reason is privacy. Service providers are increasingly concerned about disclosing the operational properties of their networks. Because of these issues, publicly available traces for analysis are limited. But thanks to NLANR [98] we have a sufficient number of traces to study the various characteristics of the fast evolving network traffic. Table 6.1 shows the traces used in this experiment. Full details of the traces are given in Appendix G.

6.2.1 Determination of the *H* parameter

The Hurst parameter of a trace is calculated in MATLAB using the Abry and Veitch method (section 2.5.2.1). A MATLAB script by the authors is used. This method is a wavelet decomposition of a time series that can be used to compute H. The input of this method is packet counts or byte counts in a small time interval (byte counts in 10 millisecond intervals are used) and the output is a wavelet-spectrum of the input series. The value of the H parameter of the input series is computed from the plot of variances of the output wavelet coefficients. The complete MATLAB script used to compute H is given in the Appendix E (section E.3).

6.2.2 Determination of the R Parameter

In all three experiments, the R parameter is sued for analysis of traffic traces. The fuzzy group model (section 5.3) is used to determine the R parameter of a traffic trace. To apply the model, a few initial parameters needed to be set. These are:

1. appropriate value for τ_G (equation (5.17))
2. membership functions of δ, B (Figure 5.11)
3. membership function of R (Figure 5.12).

By definition (equation (5.17)), if the time difference between two consecutive packets is less than or equal to τ_G, they belong to the same T-group. The value of τ_G can be set to any value by the system designer according to the design requirements. In this work, τ_G is used to be equal to the service time of a packet of the router. This means that, if the difference between time-stamps of two packets is smaller than or equal to the time required by the router to serve the packet, the two packets are in the same T-Group.

Let A be the byte count (i.e., sum of length of all the packets within the trace), P be the packet count (total number of packets within the trace) of the trace and l be the link speed, then τ_G is given by,

$$\tau_G = \frac{A}{Pl} \tag{6.1}$$

For a given trace, A, P and l are known. Therefore, using equation (6.1) τ_G can be computed. This equation shows that the value of τ_G can differ from trace to trace.

For δ and B, triangular membership functions similar to Figure 5.11 are used. The figure specifies the support and core for SMALL, MEDIUM and LARGE fuzzy sets in terms of buffer capacity β. The value of β can be specified as bytes or as a number of packets that can be buffered. For optimum performance, the buffer size should be between 10 to 100 packets [24]. The default value for all Cisco interface cards is 40 packets and for all Cisco routers are 75 packets [118]. So a buffer size of 40 to 75 packets is a reasonable choice. In this work a buffer size of 48 packets is used. To define the membership function for B (Figure 5.11(b)), the value of s is needed, where s is the time required by the router to serve one byte of data. For a given trace s is computed as:

$$s = \frac{V\rho}{A} \tag{6.2}$$

where V is the duration of the trace in seconds, A is the byte count of the trace and ρ is the utilization.

For R, the triangular membership function of Figure 5.12 is used.

Once the membership functions for δ and B are defined, R is computed using the Fuzzy Group Model (section 5.3).

6.2.3 Experimental Results and Discussion

This section contains the results of experiment 1. At first the visual summary of H and R parameters of the 84 traces used are shown. Then H and R values as a function of various traffic characteristics such as the number of connections and the link utilization of the trace are represented. The stationarity property of the traces is also studied. The results and the discussion on the results are given below.

6.2.3.1 Visual Summary of H and R Parameters

Figure 6.1 is the visual summary of the H parameter of all 84 traces of Table 6.1. Appendix G contains a summary table of all the traces used in this analysis. The interesting feature of this graph is that the H parameter is about the same for the 2003 and 2001 traces, but is lower for the 2005 traces. It has been demonstrated by several researchers that LRD originates due to the inherent TCP mechanism [5, 50]. This means that the existence of a large percentage of non-TCP (UDP and others) packets can be responsible for smaller values of the H parameter. The trace table in Appendix G shows that, in general, the 2005 traces have a higher percentage of UDP traffic than 2003 or 2001. This might be the reason for the smaller H values of the 2005 traces.

Figure 6.2 is the visual summary of the R parameter of the same 84 traces. The 2001 traces have very small R values. In general, the 2005 traces have higher R values than the 2001 and 2003 traces. The 2003 traces have higher R values than

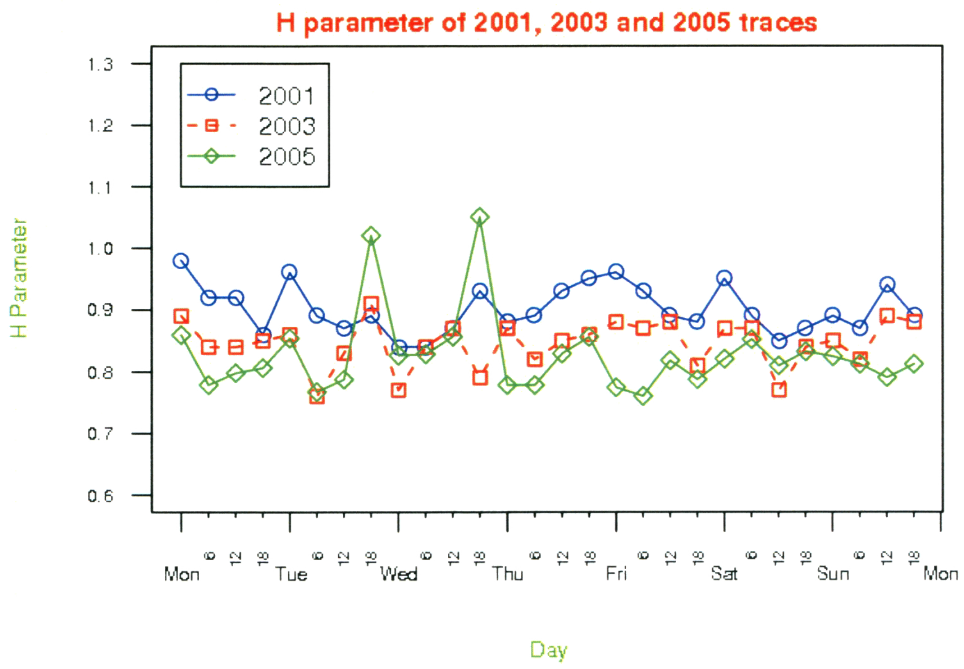


Figure 6.1: Hurst parameter of 2001, 2003 and 2003 traces at different time of the day and days of the week.

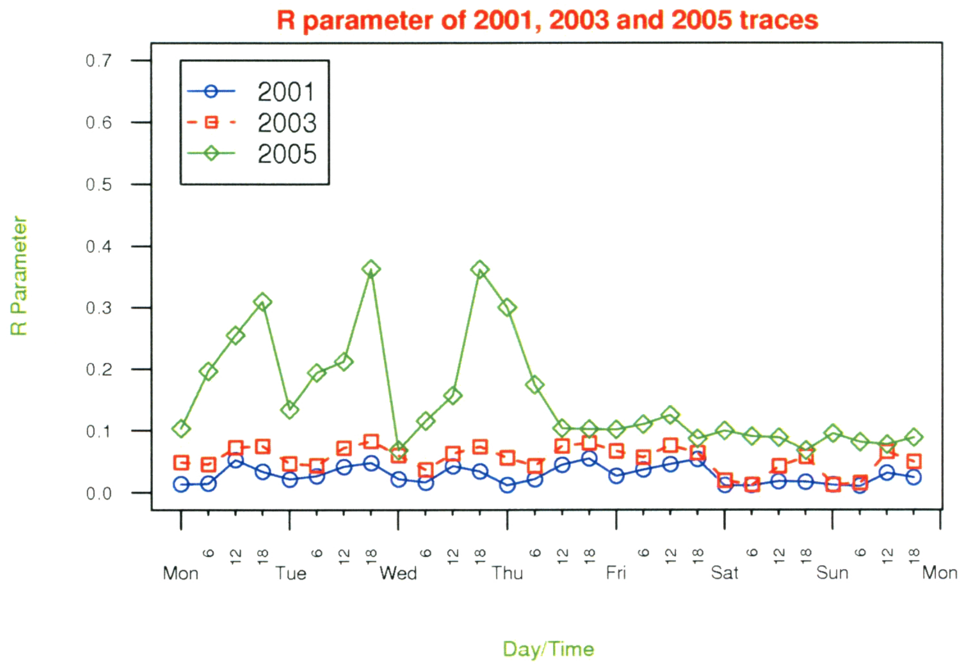


Figure 6.2: R Parameter of 2001, 2003 and 2005 traces at different time of day and days of week.

the 2001 traces. The R parameter is constructed in a way that the higher the value of R the longer the queue required (this is further studied in experiment 2). The 2001 traces have very small R values indicating that the traffic in these traces is likely to have caused little queuing on the router. The R values of the 2005 traces indicate there is likely to have been a significant amount of queuing. This is further studied in experiment 2.

6.2.3.2 H and R values at different time of day and day of week

Figure 6.1 shows that, there is no apparent relationship between the day of the week or the time of day and the H parameter. Apart from two 2005 traces, the H values during weekdays and weekends are similar. Also, H appears to be independent of the time of day.

In Figure 6.2, R is also plotted against day and time of the day for the 2001, 2003 and 2005 traces. Unlike the H parameter, the R parameter appears to be dependent on the day/time of week. In general, weekday traces have a higher R values then weekend traces, especially for 2005 traces. Apart from weekends, the midday and afternoon traces have higher R values than the night time or early morning traces. This indicates that there are likely to be longer queues during the weekday day times than weekends or night times, at least for this mix of traffic. In 2005 traces this variation in R values is higher. This indicates that the traffic pattern is becoming more and more time dependent. Therefore time-dependent load balancing or regulation might be required for optimum network performance.

6.2.3.3 Stationarity

Self-similarity is a property of a weak-sense stationary process (section 2.5.2). For self-similar traffic, the value of H can be within 0.5 to 1. Figure 6.1 shows 2 out of 28 traces from 2005 have an H value of greater than 1. This indicates that traffic is not (weakly) stationary for these two traces. The non-stationarity of these traces observed during the weekdays indicates that there might be an upward trend on the traffic byte count series. That is, the average byte count of the traffic trace at that time is becoming higher than the global average. Apart from these two, all the remaining 82 traces from 2001, 2003 and 2005 appear to be stationary ($0.5 < H < 1$).

6.2.3.4 H and R as a function of the number of connections

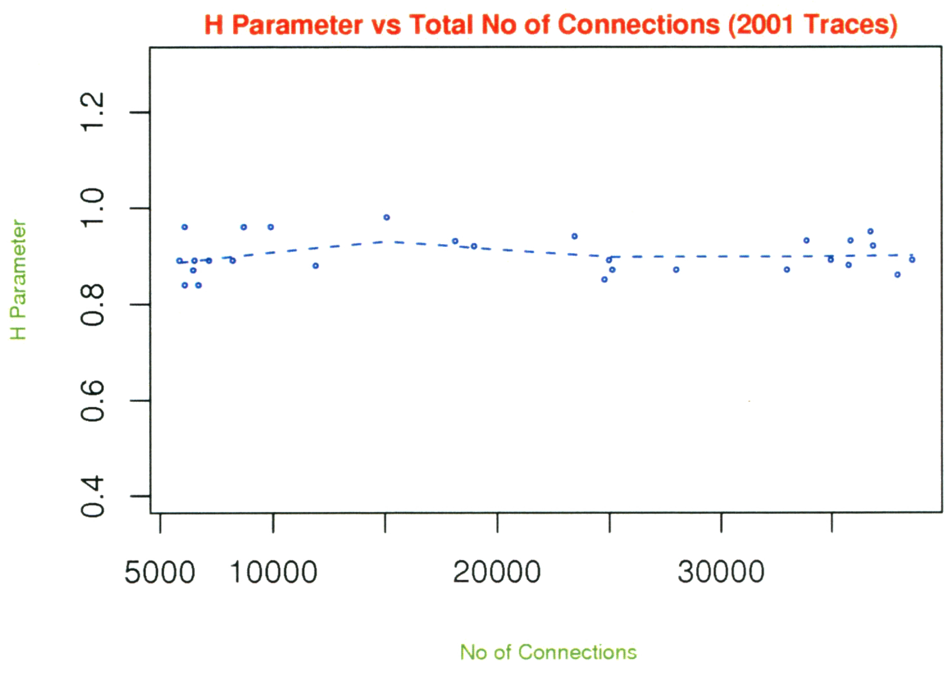


Figure 6.3: H Parameter vs Total No of Connections - 2001 Traces

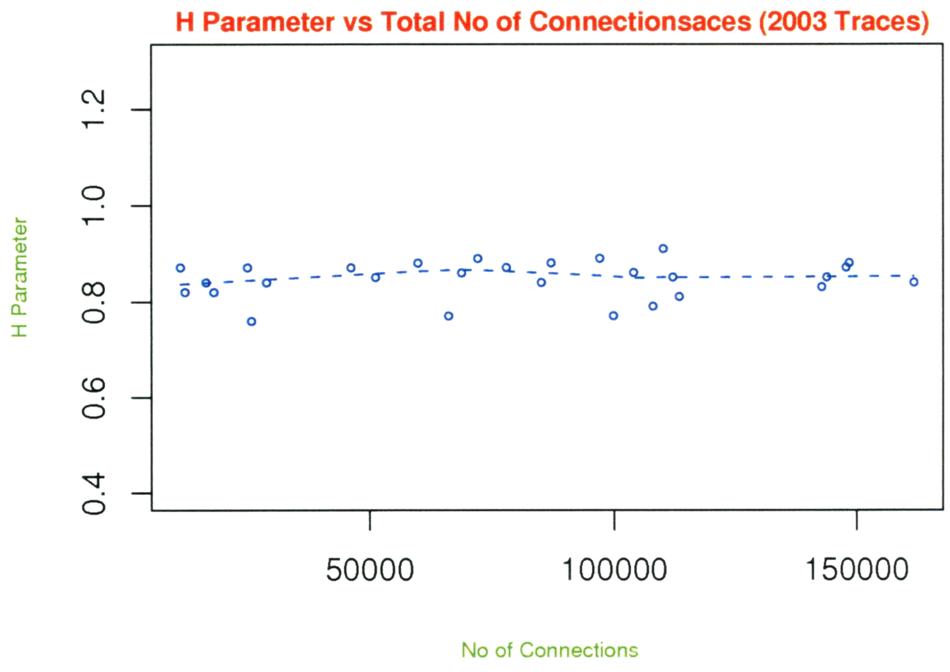


Figure 6.4: H Parameter vs Total No of Connections - 2003 Traces

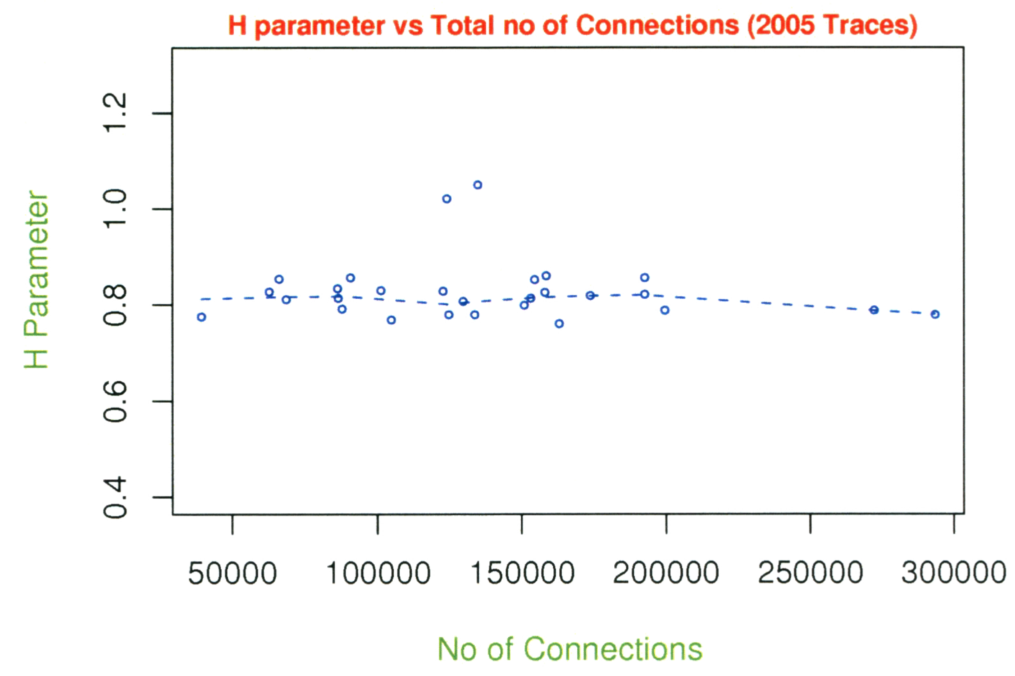


Figure 6.5: H Parameter vs Total No of Connections - 2005 Traces

Figures 6.3, 6.4 and 6.5 show the values of the H parameter as a function of the total number of connections for 2001, 2003 and 2005 traces, respectively. If the number of connections in a traffic trace is very large, due to a high level of aggregation, a non-self-similar traffic should be smooth and hence the H value might be small. That is, for non-self-similar traffic H would vary with the total number of connections. But this is not in fact the case. The figures show that the H parameter is almost independent of the total number of connections. This is true for both 2001, 2003 and even for the 2005 traces where the Internet usage is much higher. This confirms that most of the traces are self-similar. Fig 6.5 also shows the same two rogue values with $H > 1$ as Fig 6.1 for 2005.

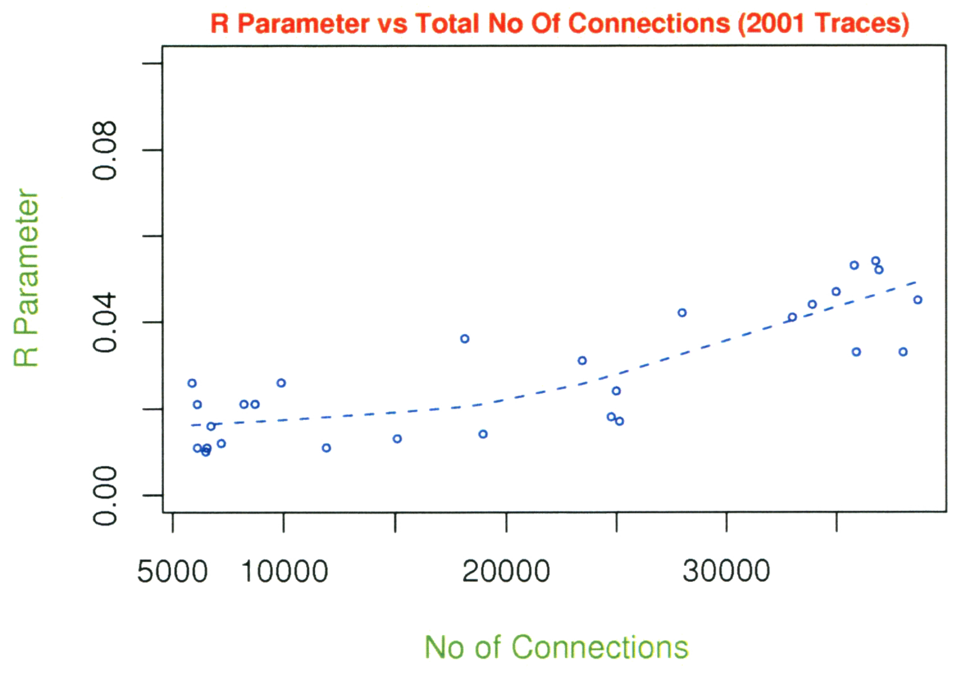


Figure 6.6: R Parameter vs Total No of Connections - 2001 Traces

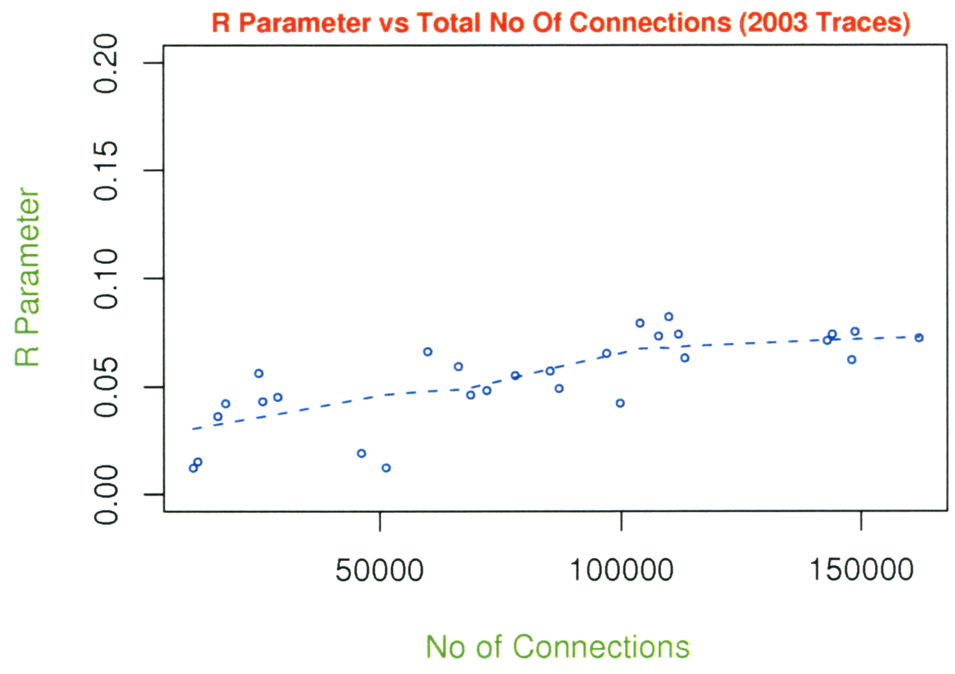


Figure 6.7: R Parameter vs Total No of Connections - 2003 Traces

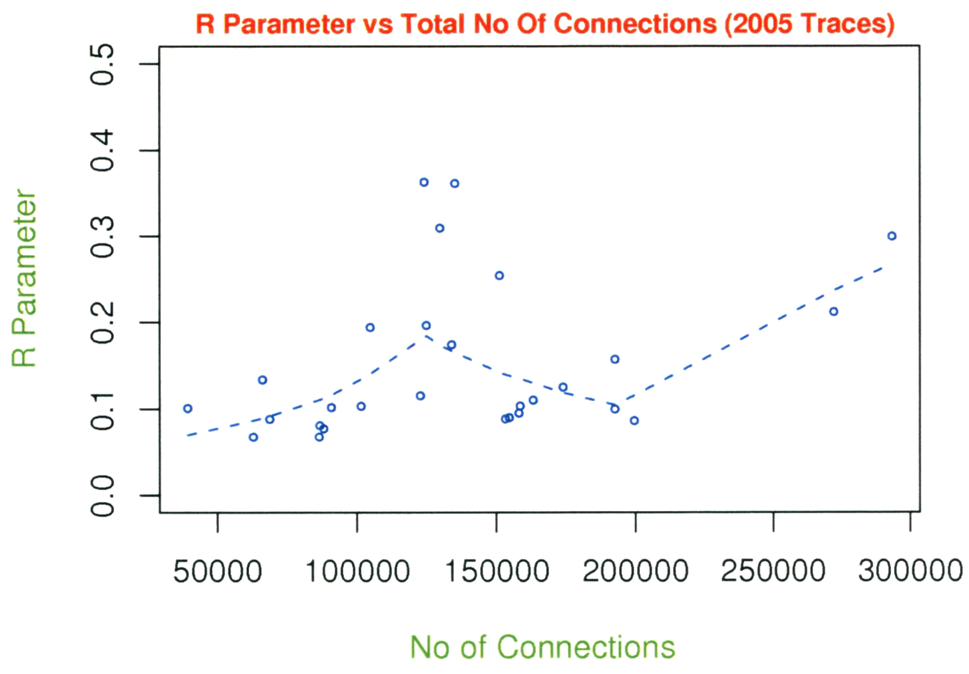


Figure 6.8: R Parameter vs Total No of Connections - 2005 Traces

Figures 6.6, 6.7 and 6.8 are plots of the R parameter against the total number of connections for 2001, 2003 and 2003 traces. It is evident that generally the R value is higher for traces with a larger number of connections for 2001 and 2003 traces. Another observation follows from the summary table in Appendix G that, a 2003 trace has a higher number of connections than a trace from 2001 with similar utilization. For example, a 2001 trace with 34,976 connections has a utilization of 5.98%. On the other hand, a trace from 2003 has 66,129 connections with a utilization of 5.56%. A possible explanation is that 2003 traces have quite a large number of UDP connections indicating the characteristics of traffic, especially transport protocol distribution is changing. The byte count of these UDP connections is generally smaller than TCP connections. Figure 6.8 shows rather inconsistent R values as a function of the total number of connections for the 2005 traces. This indicates that the relationship between number of connections and queue length shown in Fig 6.6 for 2001 traces and 6.7 for 2003 traces no longer holds true. In other words, in case of 2005, a traffic trace with a smaller number of connections can have longer queues than a trace with a larger number of connections (it will be seen in experiment 2 that higher R value require a longer queue).

6.2.3.5 H and R values as a function of link utilization

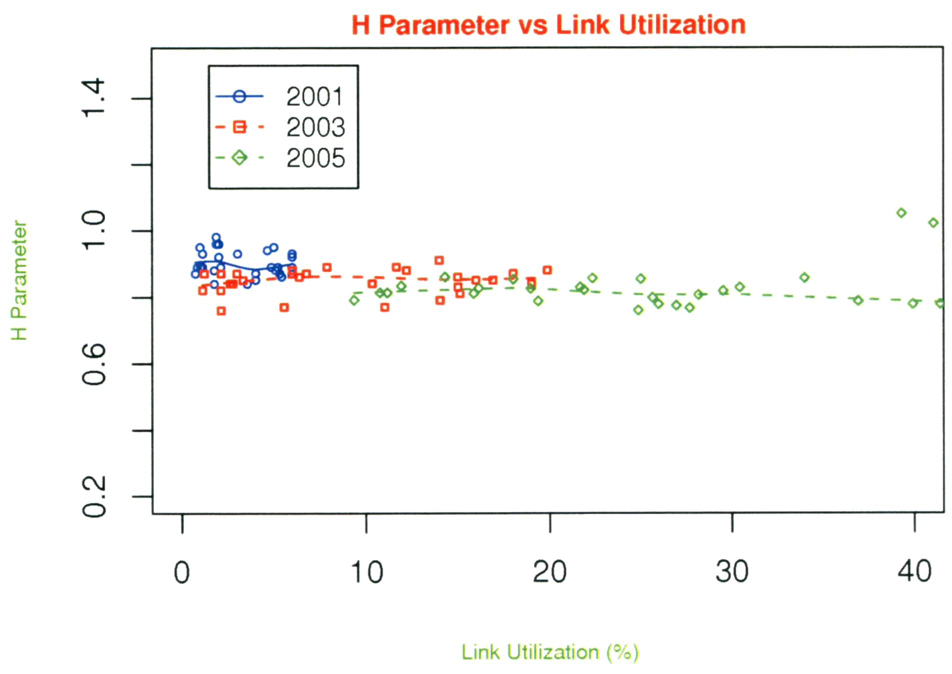


Figure 6.9: H Parameter as a Function of Link Utilization

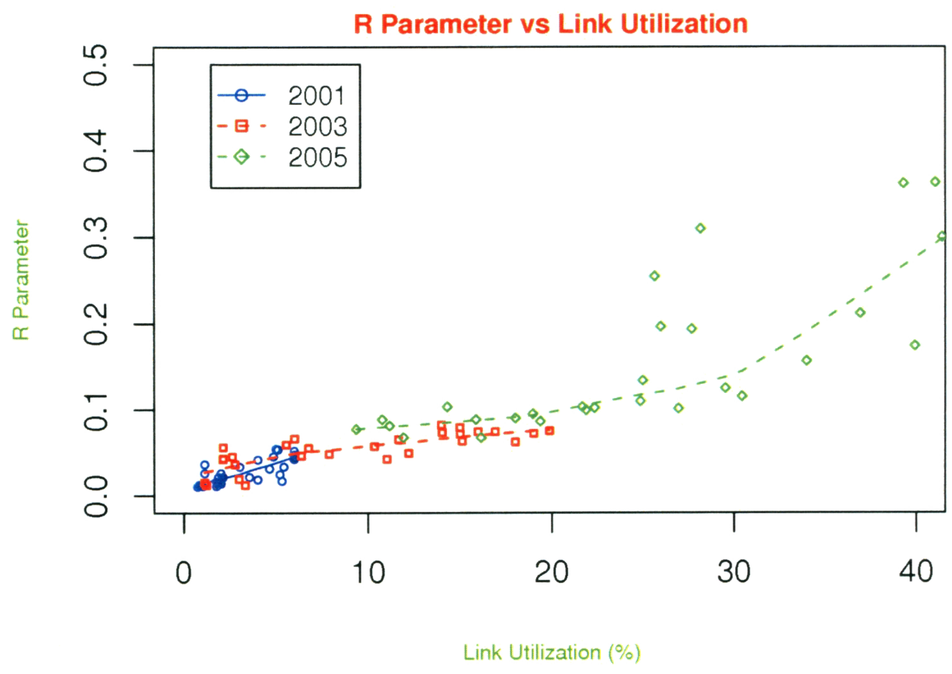


Figure 6.10: R Parameter as a Function of Link Utilization

The next set of graphs represent H and R values against link utilization. Figure 6.9 shows the H parameter appears to be independent on the link utilization for the 2001 and 2003 traces. Apart from the two traces with the hint of non-stationarity (traces with $H > 1$), H is almost independent of the link utilization for the 2005 traces. This was unexpected, as it might be thought that traffic would become smooth as the link utilization increases (section 2.7). Other research also suggests that H would increase with link utilization [79]. This is because degree of self-similarity increases as the link utilization is increased. But this is not the case for the 2001 and 2003 Auckland traces and for the AMP 2005 traces. As long as the traffic is (weakly) stationary, the H value does not seem to depend on the link utilization.

On the other hand, Figure 6.10 shows that the R parameter increases with link utilization. This is expected, as the higher utilization would cause longer queuing and hence R would be larger. Another observation is that the 2005 trace have higher R values than 2003 traces with similar utilization. That is, a 2005 trace is likely to have longer queues than a 2003 trace although they have the same utilization. A reason for this could be that the nature of traffic is changing.

6.3 Experiment 2: Analysis of Trace-Driven Queuing

In experiment 2, the queuing performance of nine real traces and two generated traces is analyzed. The aim of the analysis is to study the relationship between the H or R values with queuing performance. The queuing is simulated using csim18 [113]. The C language function that was used to implement the queue is shown in Appendix E.

For the simulation purpose traces in formats similar to the one used for calculating the Hurst parameter (a time series of byte counts aggregated in 10 millisecond time bins) are used. Using this time bins significantly reduces the number of computations without affecting the result (since bin size is small enough). A bin size of 10 milliseconds appears to be a reasonable choice as this value is used in several other trace driven queuing studies [92, 102].

6.3.1 Generating Self-Similar Traffic

In this experiment, to simulate the queuing of generated self-similar traffic, an FARIMA process (section 2.5.2.2) is used. Equation (2.14) was used to generate the self-similar series. In this equation, it can be noted that, for the n-th sample, n multiplications/additions are needed. In order to study the self-similar nature of such a process a large number of sample points are needed. As the number of multiplications/additions required to generate the n-th sample is proportional to n, a large memory and huge processing power is needed to generate such a process of large sample points. A Compaq proliant 7000 server with quad Xeon 550 processor (2 M L2 cache on each processor) and 2.5G of RAM were used to generate 60,000 points. It took about 1 hour to generate these sample points. Assuming a sample point is the number of packets arriving in a 10 millisecond time bin, the time series of 60,000 points is equivalent to a traffic trace of 10 minutes long.

The generated self-similar traffic that is used in experiment 2 is a FARIMA(0,d,0) process, where d is the fractional difference parameter (section 2.5.2.2). Two series: FARIMA(0,0.2822,0) and FARIMA(0,0.4639,0) are generated. That is, the value of d is chosen to be 0.2822 and 0.4639 respectively. They are so chosen that the two generated series have H parameter of 0.7822 and 0.9639. The queuing performances of the generated self-similar traffic with $H = 0.7822$ is compared with a real traffic trace having the same H value. A similar comparison is also made for $H = 0.9639$. These comparisons give an indication of whether the two traces (one real and one generated) with similar H values have similar queuing performance or not.

6.3.2 The Trace-Driven Queue Simulator

The simulation of queuing of a given trace is known as *trace-driven queuing simulation* [108]. Suppose a traffic trace is represented as $\{X_i,\ i = 1, 2, 3, ...N\}$ aggregated at 10 millisecond time-bins. That is, X_1 is the byte count in the first 10 millisecond interval, X_2 is the byte count in the second 10 millisecond interval and so on. Let this traffic arrive at the router with a service rate of S bytes per 10 milliseconds and infinite buffer space. After the first 10 milliseconds, the queue length will be $X_1 - S$ if $X_1 > S$; or will be zero if $X_1 < S$. That is, $Q_1 = MAX(X_1 - S,\ 0)$. This is the standard Lindley Recursion Formula [81]. According to this formula, the queue size at the n-th time interval is given by,

$$Q_n = MAX[(Q_{n-1} + X_n - S),\ 0] \quad \text{with}\, Q_0 = 0 \text{ and } n = 1, 2, ...N \tag{6.3}$$

$P(Q(t)) > L$, the Complementary Cumulative Distribution Function (CCDF) of Q is drawn against L, where, L is the queue length in bytes. The CCDF of $Q(t)$ for LRD or self-similar traffic should have "Weibull-like" decay [69]. On the other hand, as a Poisson process is memoryless, the CCDF of a Poisson process should have a sharper and faster decay. The longer the tail of the distribution, the more buffer space is required in a router to serve the traffic without dropping a packet.

The aim of experiment 2 is to study and compare the queuing behavior of traffic traces of different networks collected in different years. Equation (6.3) cannot be applied directly to simulate these traces, as the router service rate, S is unknown. An arbitrary value of S cannot be used, as router hardware might be different on different networks. A value of S that might be appropriate for a 2001 trace might not be appropriate for a 2005 trace, simply because 2005 traces have higher means and variance. A value of S that is reasonable for a 2001 data flow can lead to an unstable queue ($\rho > 1$) for the higher 2005 data flows.

We want to compare queuing performances of different traces at the same utilization level. In equation (6.3), S is defined as the maximum amount in bytes that the router can serve in 10 milliseconds. That is, if the utilization is 1, the router serves S bytes data in 10 milliseconds. S and utilization are related as:

$$\rho = \frac{E[X_i]}{S} \tag{6.4}$$

where ρ is the utilization. For a given traffic trace, $E[X_i]$ is known. Hence, substituting the value of ρ into equation (6.4), S can be determined. In these experiments, for the trace-driven queuing simulation ρ values of 0.5, 0.7 and 0.9 are used. Setting ρ to be 0.7 means, the router service rate is "scaled" so that all the traces have utilization of 0.7 [107, 108]. This chapter represents the results for ρ value of 0.7. Queuing performances of other values of ρ are given in Appendix D.

6.3.3 Experimental Results and Discussion

In this experiment trace-driven queuing simulation of nine real traces and two simulated traces are studied. For this study three real traces from each of 2001, 2003 and 2005 are chosen. The traces along with their H and R values are shown in Table 6.2. The traces are chosen so that traces of the same year have different H values but similar R values. This is because the main focus in this experiment is to study the queuing performance of traces with different H values.

Table 6.2: Traces Used to Study Queuing

Trace Name	Trace Source/Year	H	R
Auck0105	Auckland IV / 2001	0.96	0.021
Auck0106	Auckland IV / 2001	0.89	0.026
Auck0117	Auckland IV / 2001	0.96	0.026
Auck0315	Auckland VIII / 2003	0.81	0.074
Auck0319	Auckland VIII / 2003	0.77	0.075
Auck0316	Auckland VIII / 2003	0.92	0.079
AMP26	AMPI / 2005	1.02	0.362
AMP33	AMPI / 2005	1.05	0.3613
AMP22	AMPI / 2005	0.80	0.309

The Complementary Cumulative Distribution Function (CCDF) of queue length (i.e., $P(Q(t)) > L$) in 10 base logarithmic scale is drawn against the queue length. The shape of the graph is an indication of how much queuing should happen on a real router (section 6.3.2). Short queue lengths indicate little queuing delay and superior performance of the router. On the other hand a longer tail indicates a substantial amount of queuing and even some congestion indicating degraded performance.

Table 6.2 shows that the 2001 traces have higher H values than the 2003 and 2005 traces. The queuing simulation of the 2003 and the 2005 traces is of interest as both the Auck0315 and AMP22 traces have similar H values. According to the characteristics of self-similarity (section 2.5.2), the higher the value of H, the longer the tail of the $CCDF$ graph. Theoretically, traces with similar H values should have similar queuing performances.

6.3.3.1 H as an indicator of queuing performance

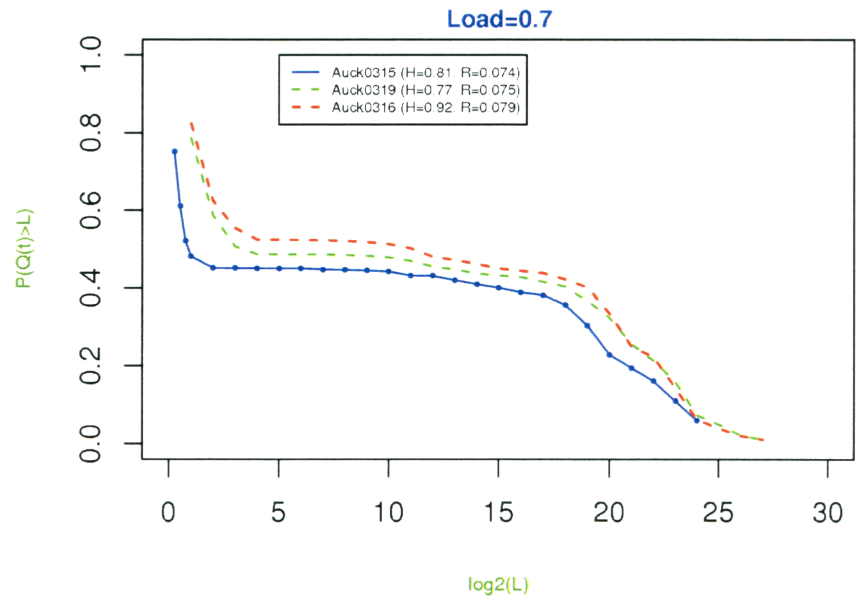


Figure 6.12: Queuing simulation of 2003 Auckland traffic

Figures 6.11, 6.12 and 6.13 show the CCDF of Q as a function of queue length for 2001, 2003 and 2005 traces respectively. The figures show that, although the 2001 traces have higher H values than the 2003 traces, their CCDF have smaller tails than the 2003 traces. One of the 2005 traces (AMP22) has similar H values as to the 2003 Auck0315 trace. But the CCDF curve of AMP22 has longer tail than that of Auck0315 trace.

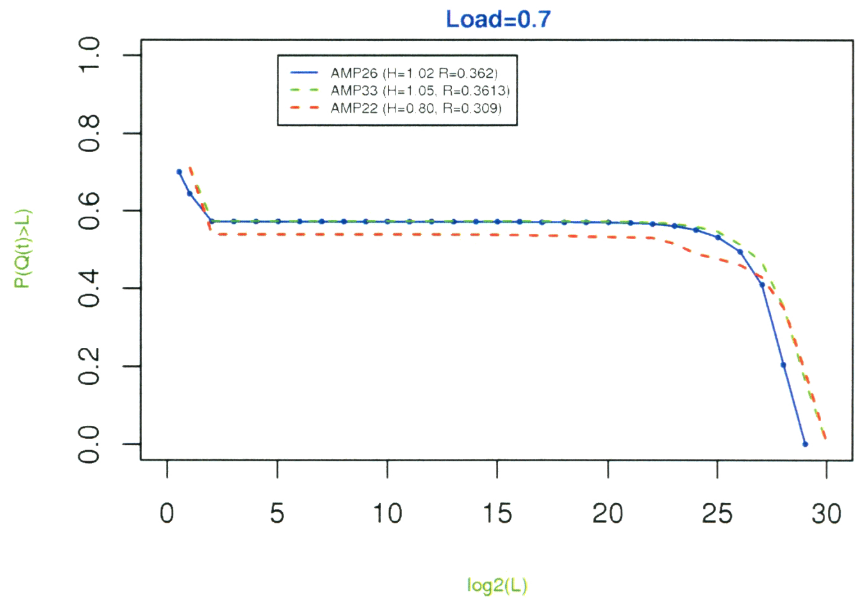


Figure 6.13: Queuing Simulation of the 2005 AMP Traffic

Figure 6.14 shows the queuing simulation of the two generated self-similar traffic data traces. They were generated using FARIMA process (section 2.5.2.2) with d equal to 0.2822 and 0.4639 respectively. These values of d generates self-similar traffic with H values of 0.7822 and 0.9639 respectively (equation (2.13)). The generated self-similar traffic with $d = 0.2822$ ($H = 0.7822$) has its H value similar to the 2003 and 2005 traces. But the comparison of Figure 6.14 to Figures 6.12 and 6.13 shows that although the generated trace has similar H values to those of the real traces, their queuing performances are not similar. Similarly, the generated self-similar traffic with $H = 0.9639$ does not have similar queuing performance to the real traffic with similar H values (2001 traces). These results indicate that a contradictory relationship exists between the H parameter and queuing performance. In other words, the H parameter by itself does not appear to be sufficient to predict queuing performance and the design of a network. Similar results have been found by other researchers [102, 107].

6.3.3.2 R as an indicator of queuing performance

The R parameter appears to be a better indicator of queuing performance. Figures 6.11, 6.12 and 6.13 show that the higher the value of R, the longer the CCDF tail. The 2001 traces have the smallest R values and their CCDF graphs have the smallest tail. Similarly, the 2005 traces have the largest R values and the tails of CCDF graphs of these traces are the longest. Also, Figures 6.11, 6.12 and 6.13 illustrate that the traces with similar R values have similar queuing performance.

Appendix D shows the queuing behavior of traces from AMP2005 with different R values and utilization of 0.5, 0.7 and 0.9. As expected, there is little queuing at $\rho = 0.5$, somewhat more at $\rho = 0.7$ and heavy queuing at $\rho = 0.9$. More importantly, the graphs which are in descending order of R values show a clear relationship between the distribution of queue sizes and R.

This suggests that the R parameter is a superior candidate for predicting queuing performance. Hence, the R parameter can be a valuable measure for network design and planning.

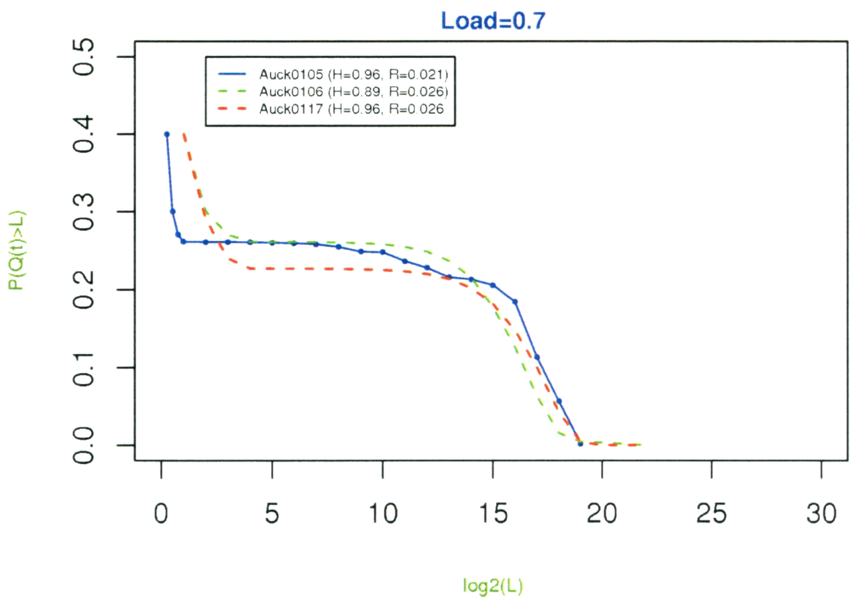
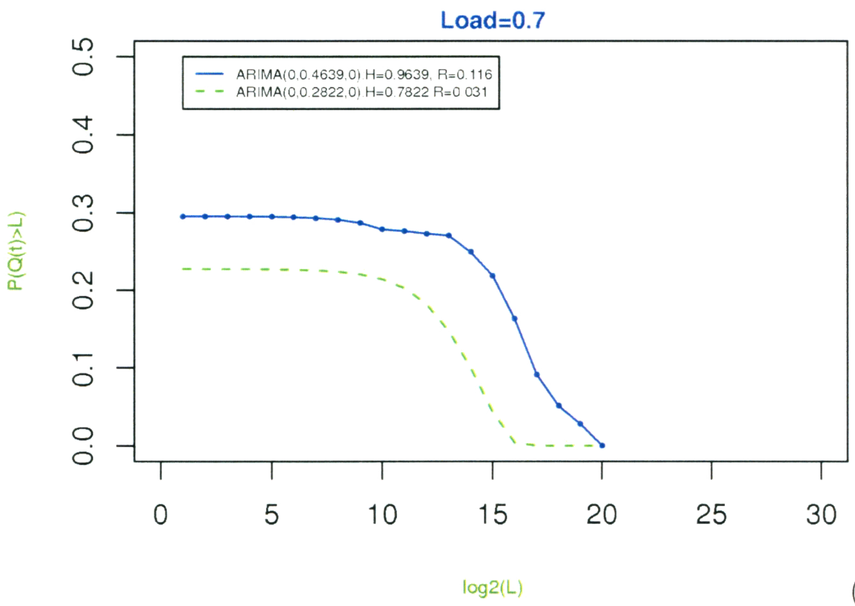


Figure 6.11: Queuing Simulation of the 2001 Auckland Traces



(b)

Figure 6.14: Queuing Simulation of Generated Self-similar Traffic.

6.4 Experiment 3: Analysis of Burstiness using Energy Plots

In this experiment, the burstiness of six real traces from 2005 are studied by plotting their energy plots (section 2.10.1).

6.4.1 Goal of the Experiment

The burstiness of a traffic trace can be measured from the energy plots of the trace (section 2.10). In section 2.5.2, it was seen that the Hurst parameter by itself characterizes self-similarity. That is the value of H indicates the degree of self-similarity. The higher the value of H, the higher the degree of self-similarity of a traffic trace. Therefore the H parameter would be expected to be proportional to the burstiness of a traffic trace. The goal of this experiment to study this hypothesis. Another goal of this experiment is to find any relationship of the burstiness of a trace with its R value.

6.4.2 Experimental Data

Table 6.3: Traces Used in the Study of Burstiness.

Trace No	Trace Name	H	R
1	AMP05-39	0.87579	0.10166
2	AMP05-38	0.88904	0.10290
3	AMP0-18	0.77846	0.19630
4	AMP05-25	0.78712	0.21176
5	AMP05-33	1.05011	0.36130
6	AMP05-26	1.02001	0.36235

The traces used along with their H and R values are shown in Table 6.3. The traces are chosen in a way that they have a range of R values. Traces 1 and 2 are chosen to have similar R values. Similarly, traces 5 and 6 are chosen to have similar R values. This is because we want to examine whether traces with similar R values have similar energy or not. Similarly trace 3 and trace 5 have similar H values. The energy plots of these traces should give a clear picture of relationship of burstiness with H and R.

6.4.3 Experimental Methods

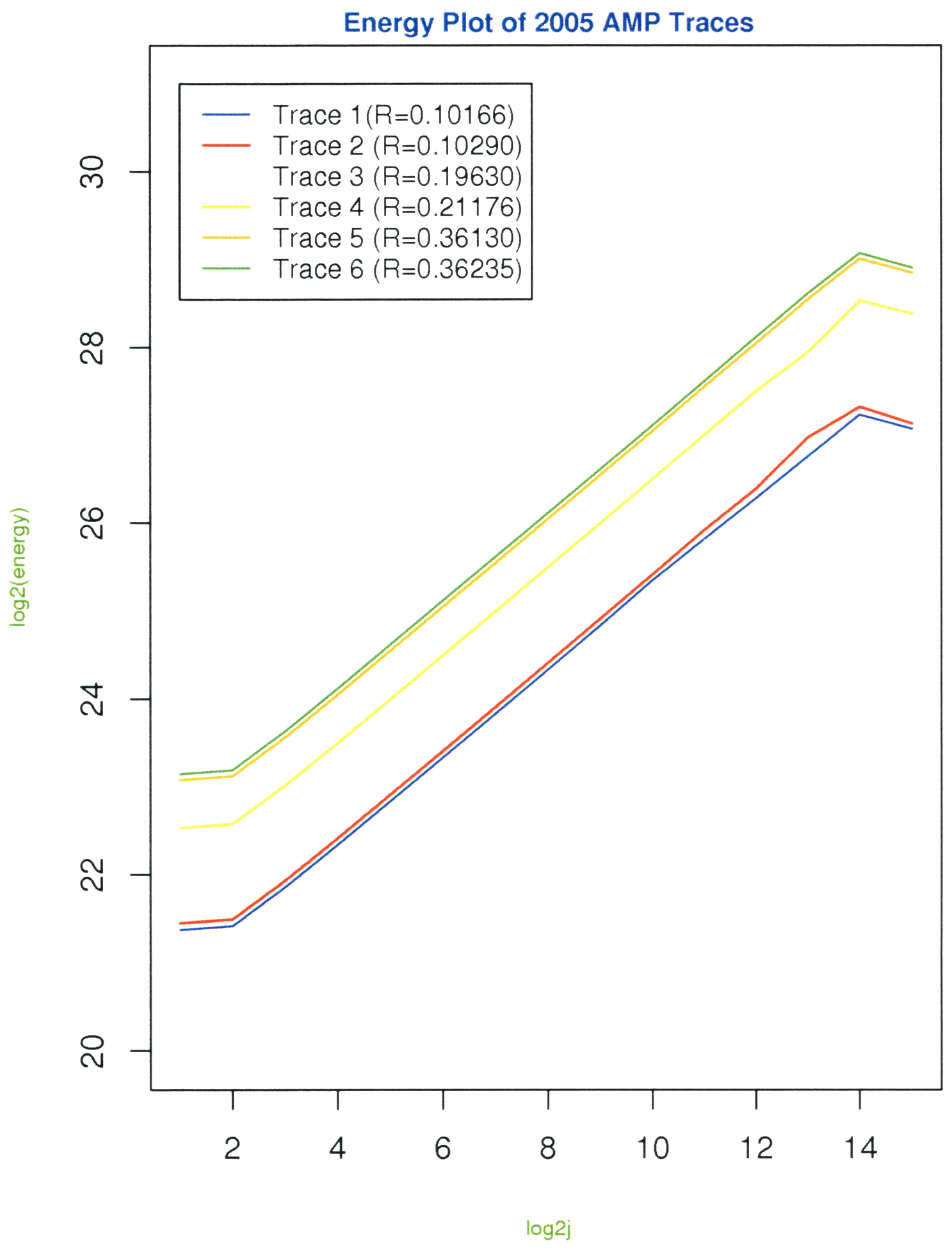


Figure 6.15: Energy Plots of the Traces of Table 6.3

Energy plots can be indicators of the burstiness of a traffic trace (section 2.10). Bursty traffic has higher energy than smooth traffic. Although energy plots are used mainly to study the behavior of traffic, in this work energy plots are used to capture the burstiness of the traffic. The energy plots are plotted using a method similar to [70] where the magnitude of the energy of a traffic trace is computed and is drawn against scale j.

6.4.4 Results

Figure 6.15 shows the energy plots of the six traces used. The results are discussed below.

6.4.5 *H* as a measure of Burstiness

According to Table 6.3, trace 3 and trace 4 have similar H values. Figure 6.15 shows that although they have similar H values, their energy levels are different. Trace 4 has much higher energy than trace 3. Higher energy means burstier traffic. Trace 1 has higher H value than trace 4. But trace 4 has higher energy than trace 1. Therefore, the H parameter appears not to be a measure of burstiness.

6.4.6 *R* as a measure of Burstiness

Among the traces used trace 1 and trace 2 have very similar R values (Table 6.3). Figure 6.15 shows that these traces also have very similar energy levels. Trace 5 and trace 6 have very close R values. The plot shows that they also have similar energy levels. Trace 6 has the largest R value. It also has the highest energy. The R values of all 6 traces are consistent with their energy plots. The plots shows that, traces with similar R values have similar energy and a trace with higher R value has higher energy than that of traces with smaller R values. Therefore, the R parameter appears to be consistent with energy plots and appears to be a good measure of burstiness of traffic.

6.5 Conclusion

This chapter analyzed 84 real traces. The H and the R values of these traces were computed . The relationship of these values with different statistical properties of the traces were studied. Queuing performance and burstiness analysis using MRA energy plots are also been conducted. It was found that the H parameter alone may not be sufficient be used to predict queuing performance. On the other hand, the R parameter appeared to be a reasonably accurate estimator for queuing performance. The R parameter also appeared to be useful for measuring the burstiness of a traffic.

Chapter 7

Router Performance Optimization Using Our Proposed Traffic Model

This Chapter focuses on how the proposed traffic models can be used for optimization of a router's queuing performance. The performance of a router depends heavily on both buffer dimensioning and queue management. Some of the key buffer dimensioning issues and Active Queue Management (AQM) schemes were addressed in Chapter 3. In section 3.5, router performance optimization was formulated as a multi-objective optimization problem. In section 3.6 the design principles for an ideal AQM scheme were discussed. Following these design principles, in this Chapter an AQM scheme is proposed. This is named *Fuzzy Adaptive Fair Random Early Detection* (FAFRED).

7.1 FAFRED (Fuzzy Adaptive Fair Random Early Detection)

FAFRED is our proposed AQM scheme based on the fuzzy state model (section 5.2) and fuzzy group model (section 5.3). The aim of FAFRED is to provide a fair AQM scheme with optimum performance. For optimum performance, it needs to calculate a proper packet drop probability in a non-biased manner. The drop probability is computed based on the current congestion state and the current packet arrival rate. FAFRED does this by determining the current fuzzy state of the traffic using the fuzzy state model (section 5.2) . FAFRED computes the degrees of membership of linguistic variable STATE to the corresponding fuzzy states and based on these membership values, it computes the *global drop probability* $r(t)$ at time t.

In order to make the packet dropping fair, FAFRED then computes the R parameter of each flow and the actual drop probability of the packet is computed based on the global drop probability and the R values of all the active flows as described below. This work calls the actual packet drop probability the *packet drop probability*. If the packet selected to be dropped is ECN capable (section 3.3), it can be marked instead of being dropped. The packet drop probability of packet p_i is the actual drop probability of the packet and is denoted by r_{p_i}, which is expressed as a fraction. In FAFRED, a random number between 0 and 1 is generated. If the generated number is less than or equal to r_{p_i}, packet p_i is dropped. The block diagram of the scheme is shown in Figure 7.1.

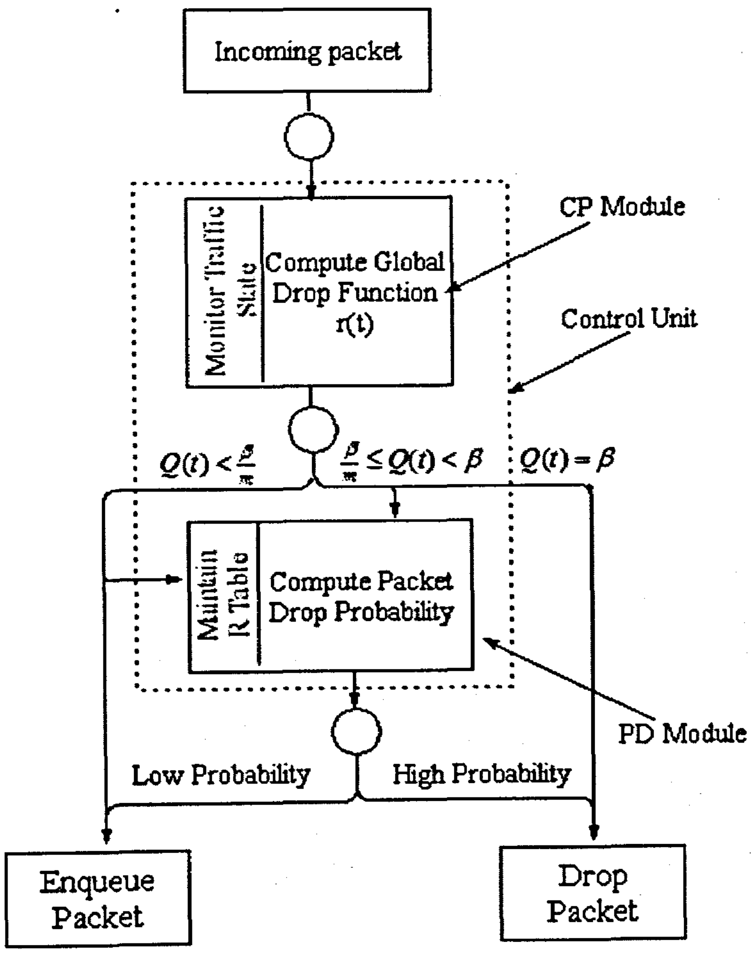


Figure 7.1: Block diagram of FAFRED. The CP module predicts congestion by monitoring traffic state and computes global drop function $r(t)$. The PD module does the actual dropping.

In FAFRED, dropping a packet involves two steps: i) calculation of the global drop probability and ii) calculation of the packet drop probability. These two are handled by the two sub-modules. The *Congestion Prediction* (CP) module calculates the drop probability and the *Packet Drop* (PD) module calculates the packet drop probability and does the actual dropping.

The Congestion Prediction (CP) Module

The CP module monitors continuously the arrival of packets. Based on a fuzzy IF-THEN rule-base it computes the global drop probability $r(t)$. The rule-base used in this work is shown in Table 7.1. The CP module constantly computes and monitors the instantaneous queue length, $Q(t)$ and the variable STATE (as described in section 5.2). Whenever it matches any of the rules of Table 7.1, it calculates the global drop probability and passes this value to the PD unit. If there is no match to a rule, the drop probability remains unchanged. Similarly to SRED (equation (3.6)), FAFRED uses a step drop function. That means, the drop probability can increase or decrease only in steps. In FAFRED, the global drop probability can increase (or decrease) in steps only by amount η or ϵ. These are the two initial parameters of FAFRED. The procedure to adjust the values of η and ϵ is described later in this section.

Table 7.1: Rule-base used by the CP module to compute the global drop probability.

No	Condition	$r(t)$
1	$Q(t) < \frac{\beta}{m}$	0
2	$Q(t) = \beta$	1
3	STATE is QUIET	0
4	STATE is BURST	$\min(1, r(t-1) + \eta)$
5	STATE is UPWARD ALARM	$\min(1, r(t-1) + \epsilon)$
6	STATE is ALARM	$\min(1, r(t-1) + \epsilon)$
7	STATE is GREEDY	$\min(1, r(t-1) + \eta)$
8	STATE is DOWNWARD ALARM	$\min(1, r(t-1) - \epsilon)$
9	STATE is DOWNWARD QUIET	0
10	STATE is CONSTANT	$r(t-1)$

In Table 7.1, β is the buffer capacity, m is a positive integer and $r(t)$ is the global drop probability. The first rule says, if the current queue length is less than $\frac{\beta}{m}$, no packet is dropped. Rule 2 says, if the queue is full, then drop the incoming packet. According to rule 3, if the STATE is QUIET, the global drop probability is zero. This means FAFRED does not drop any packet when the STATE is QUIET. Similarly, according to rule 4 and rule 5, the drop probability is increased by amount η when STATE is BURST and the drop probability is increased by an amount ϵ when the STATE is UPWARD ALARM.

The Packet Drop (PD) Module

The PD module receives the global drop probability from the CP module. The PD module computes and maintains the R values of all the active flows. These R values are stored in a table. This work calls this table the "R Table". Suppose at time t, a packet p_i of flow f_i arrives at the router. The drop probability r_{p_i} of the packet p_i is computed as:

$$r_{p_i} = \frac{r(t)R_{f_i}}{E(R_{f_j})} \tag{7.1}$$

where R_{f_i} is the R value of the flow f_i and $E(R_{f_j})$ is the average of the R values of all the active flows. If p_i is dropped, the value of R_{f_i} is equated to zero and the corresponding entry in the R Table is updated.

Algorithm 1 Algorithm for the PD Module

```
IF $r(t) = 0$
    Enqueue $p_i$
ELSE
    COMPUTE packet drop probability $r_{p_i}$
    Generate a random number between 0 and 1
    IF the random number is greater than $r_{p_i}$
        Enqueue $p_i$
        SET $R_{f_i} = 0$
    ELSE
        Drop $p_i$
        COMPUTE $R_{f_i}$
    ENDIF
ENDIF
UPDATE $R_{f_i}$ in R Table
```

The algorithm for the PD module is shown in algorithm **1**. The PD module drops a packet only when the global drop probability, $r(t)$ is greater than zero. Equation (7.1) shows that the packet drop probability for packet p_i, r_{p_i} is proportional to the R_{f_i}. That is, a packet from a flow with a high R value is more likely to be dropped than a packet from a flow with a small R value.

7.2 Discussion of FAFRED

Ideally an AQM should be as fair as possible and exhibit optimum performance according to the properties of l, q, u and v (Section 3.5.2). FAFRED tries to maintain a smaller queue length by dropping the minimum number of packets. Equation (7.1) shows that the drop probability for the packet p_i is directly proportional to R_{f_i}. According to the characteristics of the R parameter, a flow with a higher R value is likely to cause more queuing in the router. Thus selecting a packet for dropping from a flow with the highest R value should lead to the queue length being maintained to a smaller level by dropping the minimum number of packets.

Once the packet p_i is dropped, R_{f_i} is set to zero. As the drop probability for the packet p_i is proportional to R_{f_i}, this equates the drop probability for the packets of flow f_i to zero. The other flows have non-zero R values and hence a higher drop probability. When the next packet is dropped, it is likely that a packet from another flow is selected for dropping. In this way, FAFRED achieves fairness.

7.2.1 Discussion on Setting up Parameters for FAFRED

FAFRED requires 3 initial parameters: m, ϵ and η. In Table 7.1, β is the buffer capacity, m is an integer and $Q(t)$ is the current queue length.

The objectives for our AQM are to minimize l, q, $(-\rho)$ and v (Section 3.5.2). In this work the aggregating approach (Section 4.2.3.1) is used, where multiple objectives are combined into a single objective function by means of aggregation or addition. Applying the aggregating functions method (equation (4.5)) and using $f_1(x) = l$, $f_2(x) = q$, $f_3(x) = -\rho$ and $f_4(x) = v$, the optimization of an AQM performance requires minimizing the following objective function:

$$F(l,q,\rho,v) = w_1 l + w_2 q - w_3 \rho + w_4 v \tag{7.2}$$

Equation (7.2) shows that to optimize the performance of the router we need to minimize F which is a function of l, q, ρ and v. The parameters l, q, ρ and v depend on $r(t)$ and $r(t)$ is computed based on m, ϵ and η (Table 7.1). That is, l, q, ρ and v are themselves (unspecified) function of m, ϵ and η. Therefore, no mathematical formulae on how l, q, ρ and v depends on m, η and ϵ can be written. In other words, F in equation (7.2) is a complex nonlinear function of m, η and ϵ. An approach to optimizing nonlinear functions is to use a Genetic Algorithm (GA) as it does not require any mathematical function or derivative to optimize as required by other methods [53]. Hence a GA is sued to find the optimum value of m, η and ϵ for which F is minimum. As l, q, ρ and v are themselves (unspecified) function of m, ϵ and η, the objective function of equation (7.2) becomes

$$F(m,\eta,\epsilon) = w_1 l(m,\eta,\epsilon) + w_2 q(m,\eta,\epsilon) - w_3 \rho(m,\eta,\epsilon) + w_4 v(m,\eta,\epsilon) \qquad (7.3)$$

Previous research works are studied to determine m. The original RED does not drop any packets when buffer occupancy is less than MIN_{th} (equation (3.5)). In [101], Ott showed that an optimum value for MIN_{th} is $\frac{\beta}{6}$, where β is the buffer capacity. Similarly to RED, SRED starts to drop packets when the buffer reaches $\frac{\beta}{6}$. Therefore, in the case of a simple network $m = 6$ can be assumed. For more complex live networks, it may be necessary to explore other values of m in search of the optimum result.

According to Table 7.1, $r(t)$ is increased by amount η when the traffic is in BURST and is increased by amount ϵ when the traffic state is ALARM or UPWARD ALARM. As the drop probability in the BURST state should be higher than the drop probability in the ALARM state, it is logical to set the condition $\eta > \epsilon$. In the fuzzy group model, the membership functions of δ and B (Figure 5.11) are constructed in such a way that the core of the LARGE fuzzy set is 3 times the core of the SMALL fuzzy set. The original RED also suggests that MAX_{th} be approximately 3 times MIN_{th}. Therefore, it is decided in this work to use $\eta = 3\epsilon$. Floyd, the author of the original RED showed that for optimum performance, the queue drop probability should not exceed 2% [45, 46]. Therefore, η can be restricted to be less than 0.02 (i.e., $\epsilon < 0.0067$).

Assuming $m = 6$ and $\eta = 3\epsilon$, the GA can be simplified to find the value of ϵ for which F is minimum. Hence, in this special simplified case FAFRED requires,

$$\begin{aligned} &\textit{minimize} \quad F(\epsilon) \\ &\textit{subject to,} \\ &\qquad \epsilon < 0.0067 \\ &\qquad \eta = 3\epsilon \\ &\qquad m = 6 \end{aligned} \tag{7.4}$$

Equation (7.4) shows that setting up the parameter for FAFRED requires setting up a GA to get a value of ϵ for which the solution of equation (7.4) is Pareto optimal. The assumptions $m = 6$ and $\eta = 3\epsilon$ reduce the complexity of GA, but they might not give the optimum performance in a real complex network. In a complex network, the GA should be set up to find the optimum values for m, η and ϵ though this is out of the scope of this research. However, a simple GA to find a feasible solution of equation (7.4) in the FAFRED simulation study is set up (section 7.3.4).

The weights, w_i depend on the performance goal of the network. For example, if the design goal is to achieve higher utilization, w_3 can be set higher than other weights.

7.3 Simulation of FAFRED and Other AQM Schemes

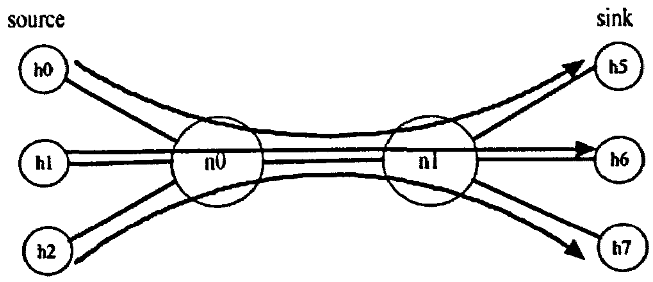


Figure 7.2: Topology Used in the Simulation of various AQM scheme.

The simulation package J-Sim [95] is sued to simulate FAFRED and to compare its performance with other AQM schemes. The throughput vs time, the TCP sequence number vs time and the queue length vs time are plotted for each of RED, FRED, SRED, drop-tail and FAFRED. The fairness index for each of these

AQM schemes are also computed. The results are discussed in section 7.3.5. The pseudocode for the simulation using FAFRED is shown in Appendix E.

7.3.1 Topology Used

The simple topology of Figure 7.2 is used for the simulation. The network consists of three sources ($h0$, $h1$ and $h2$), three sinks ($h5$, $h6$ and $h7$) and two routers ($n0$ and $n1$). The three sources and the three sinks are connected to the routers by 10 Mbps (10×2^{20} bytes per second) links. The link between the routers $n0$ and $n1$ is a 10 Mbps bottleneck link with a buffer for 20 packets. A bottleneck link is an overloaded link that can be congested.

7.3.2 Simulation Setup

To run the simulation, three TCP connections are established in sequence. For each AQM, the simulation runs for 1200 seconds. Let the connections between $h0$ and $h5$, $h1$ and $h6$ and $h2$ and $h7$ be labeled f_0, f_1 and f_2 respectively. At $t = 0$, f_0 is established and remains the only active flow until $t = 50$. At $t = 50$, f_1 is established. Both f_0 and f_1 are active until $t = 100$. At $t = 100$, f_2 is established. During $t = 100$ to $t = 600$, all three connections are active. At $t = 600$, f_0 stops. Both f_1 and f_2 are active from $t = 600$ to $t = 1000$. At $t = 1000$, f_1 stops. From $t = 1000$ to $t = 1200$, f_3 is the only active flow (Table 7.2). A J-Sim built in flow generation method that produces the same packet distributions all the time is used. That is, the same network and the same traffic conditions are used to study all the AQM schemes.

7.3.3 Initial Parameters

Simulation of all the AQM schemes requires the buffer size β. In this simulation, for all the AQM schemes, $\beta = 20$ packets is used. That is the buffer can queue

Table 7.2: Active Connections During FAFRED Simulation

Time	Active Connections
0-50	f_0
50-100	f_0, f_1
100-600	f_0, f_1, f_2
600-1000	f_1, f_2
1000-1200	f_2

up to 20 packets.

J-Sim has built in classes for each of RED, SRED, FRED and drop-tail. Each built-in class has a default value for all the initial parameters. In this simulation, apart from β, these default values for each of these AQM schemes are used.

For each AQM the simulation ran for 20 minutes.

7.3.4 Setting up the GA for FAFRED Optimization

Initially the router operates in *training mode* to get the optimum value for ϵ. Before running the GA we have to scale l, q, ρ and v and set up the weights of the aggregating function F (Section 4.3). In the case of this simple simulation v can be ignored. This is because small values of q and l, will keep v to an acceptable minimum value. Although, again in the case of a complex real network it might be unwise to ignore v.

Section 4.3 discussed the scaling and setting up of weights of an optimization problem. In this simulation the weights are scaled in a way that all the objectives can have values between 0 and 1.

The range of l is set to be $[0, 0.02]$. That is, a packet loss rate of 0.02 or higher is equivalent to $l' = 1$, where l' is the scaled value of l. A loss rate of 0 is equivalent to $l' = 0$. A loss rate between $[0, 0.02]$ has the scaled value $\frac{l}{0.02}$.

In this simulation the buffer capacity is assumed to be 20 packets. That is, q can be in $[0, 20]$. The scaled value of q is given by $q' = \frac{q}{20}$.

Utilization ρ can have values in $[0, 1]$. However, a utilization of < 0.7 could considered to be too low for this simple simulation where there can a maximum of three active connections. Hence the threshold of ρ is set to be $[0.7, 1]$. A utilization of 0.7 has the scaled utilization $\rho' = 0$ and a utilization of 1 has a scaled value of 1. Any value of ρ in $[0.7, 1]$ has the scaled value $\rho' = \frac{\rho - 0.7}{0.3}$.

Once the scaling is performed, FAFRED requires the optimization of the following function

$$F(\epsilon) = w_1 l' + w_2 q' - w_3 \rho' \qquad (7.5)$$

The next step is to set up the weights w_1, w_2 and w_3. As mentioned in section 4.3, setting up proper weights depends on the network. In the case of this simulation, no particular objective is dominant. Therefore, it is assumed that $w_1 = w_2 = w_3 = 0.33$.

The next step is to set up the GA parameters and run the GA. In this step at first it is required to encode ϵ on a chromosome. A chromosome is chosen to be a random bit string of length 10. It is assumed ϵ to be in [0, 0.0067] (Section 7.2.1). Therefore, a bit string 1111111111 is equivalent to $\epsilon = 0.0067$ and a bit string of 0000000000 is equivalent to $\epsilon = 0$. For other values, ϵ is encoded as $\frac{\text{decimal value of the chromosome}}{2^{10}} \times 0.0067$. In the case of training FAFRED for this simple simulation we have assumed the following GA parameters:

- Population size = 4
- Crossover rate = 0.7
- Number of crossover point = 1
- Crossover type = one point crossover
- Mutation rate = 0.1
- Maximum number of generation = 5

The training follows a procedure similar to Appendix C. Each chromosome of the initial population is a candidate solution for the value of ϵ. For each candidate value of ϵ, the router operates using FAFRED for a specific amount of time and calculates the fitness of equation (7.5). After each generation, new population is formed and the router continues to operate using FAFRED with a candidate value of ϵ. Once the termination condition (5 generations) is met, the value for ϵ is found.

The GA parameters that used here are too simple to be applied on a real network. But as the router being trained is to operate in a simple topology where there can be a maximum of three flows, these parameters might just be sufficient to produce a non-dominant solution.

The topology of Figure 7.2 ran in training mode for 100 minutes. Each generation ran for 20 minutes. After 5 generations, the training stopped. The value for ϵ for the topology of Figure 7.2 was found to be 0.00513. That is, for FAFRED, the parameters are:

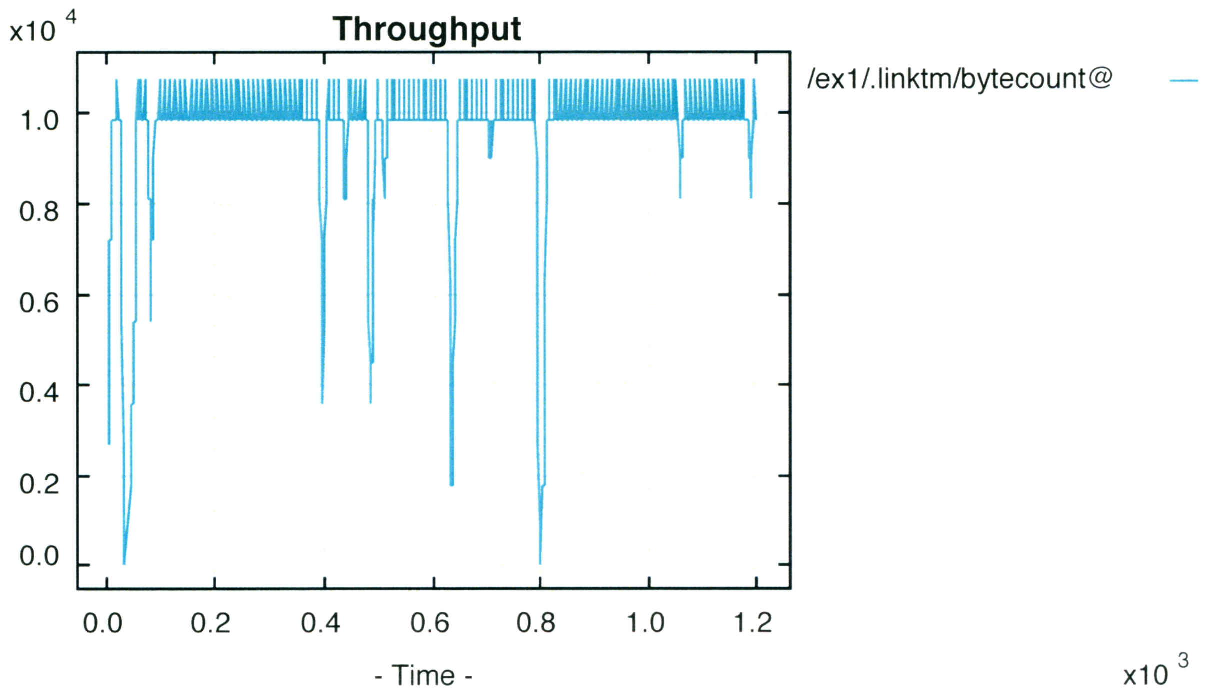
$$\begin{aligned} m &= 6 \\ \epsilon &= 0.00513 \\ \eta &= 0.01539 \\ \beta &= 20\,packets \end{aligned} \tag{7.6}$$

Once the initial parameters are found, the simulation ran for 20 minutes using FAFRED.

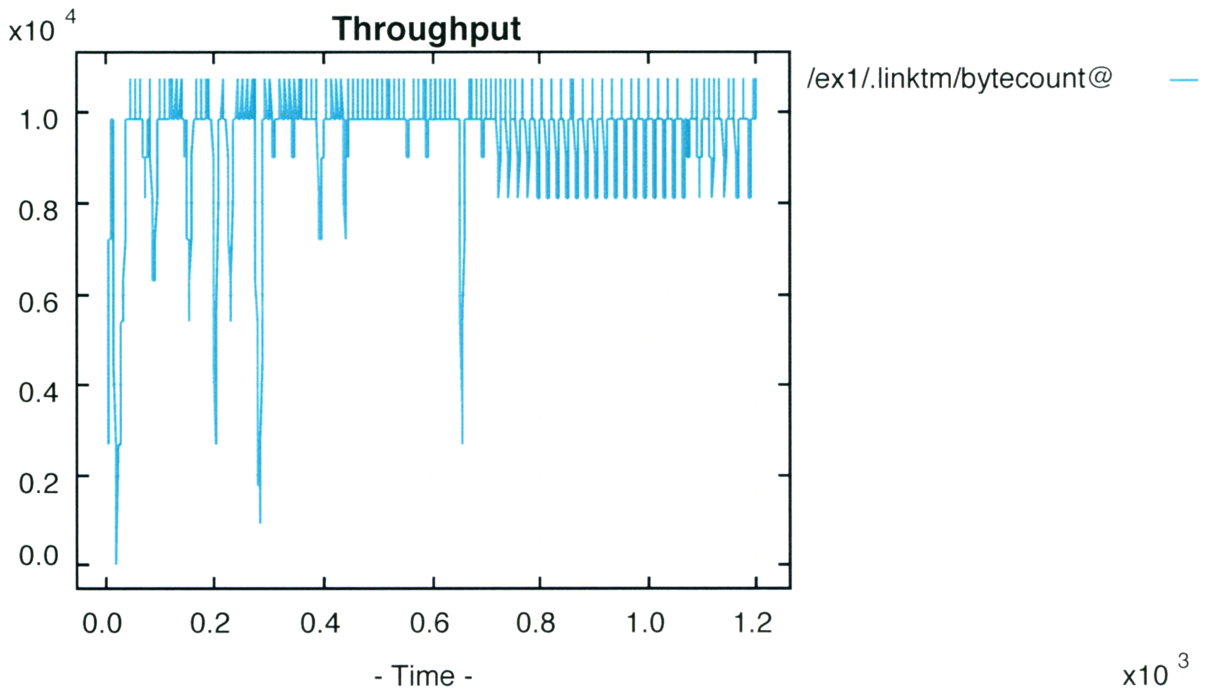
7.3.5 Analysis of the Results

7.3.5.1 The throughput vs time plots

Figures 7.3 and 7.4 show the throughput of bytes per second of all the AQMs studied. Between $t = 100$ to $t = 600$, when all three flows are active, FAFRED has the highest and the most stable throughput. There are some sharp drops of throughput in RED and in FRED indicating the utilization is not close to the maximum. FAFRED avoided such a drop during the entire period when all 3 flows were active. Drop-tail also has very high throughput.

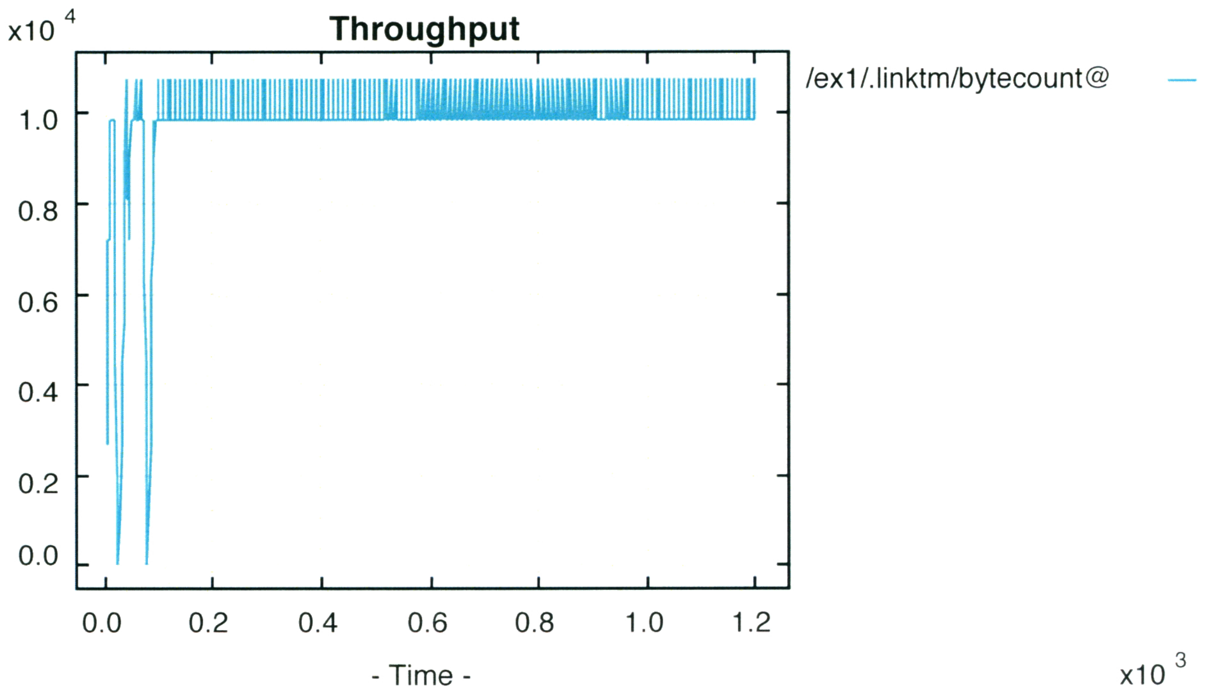


(a)

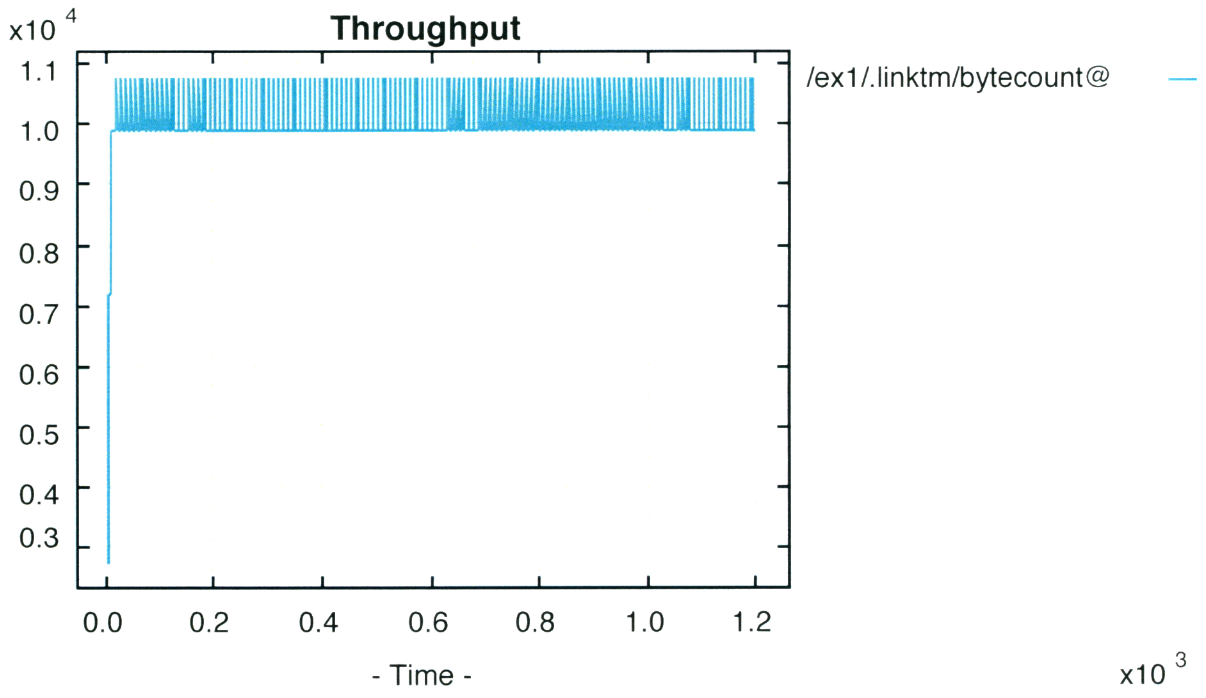


(b)

Figure 7.3: Throughput (in bytes per second) vs Time (in second) Plots of (a) RED (b) FRED



(a)

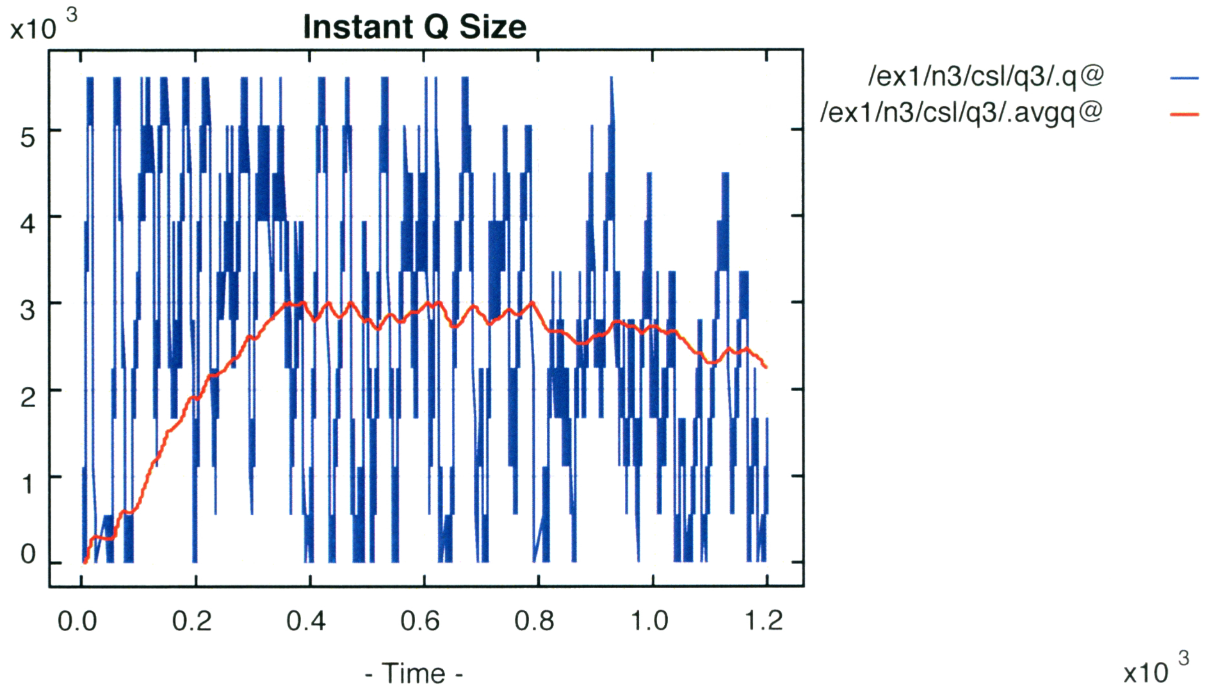


(b)

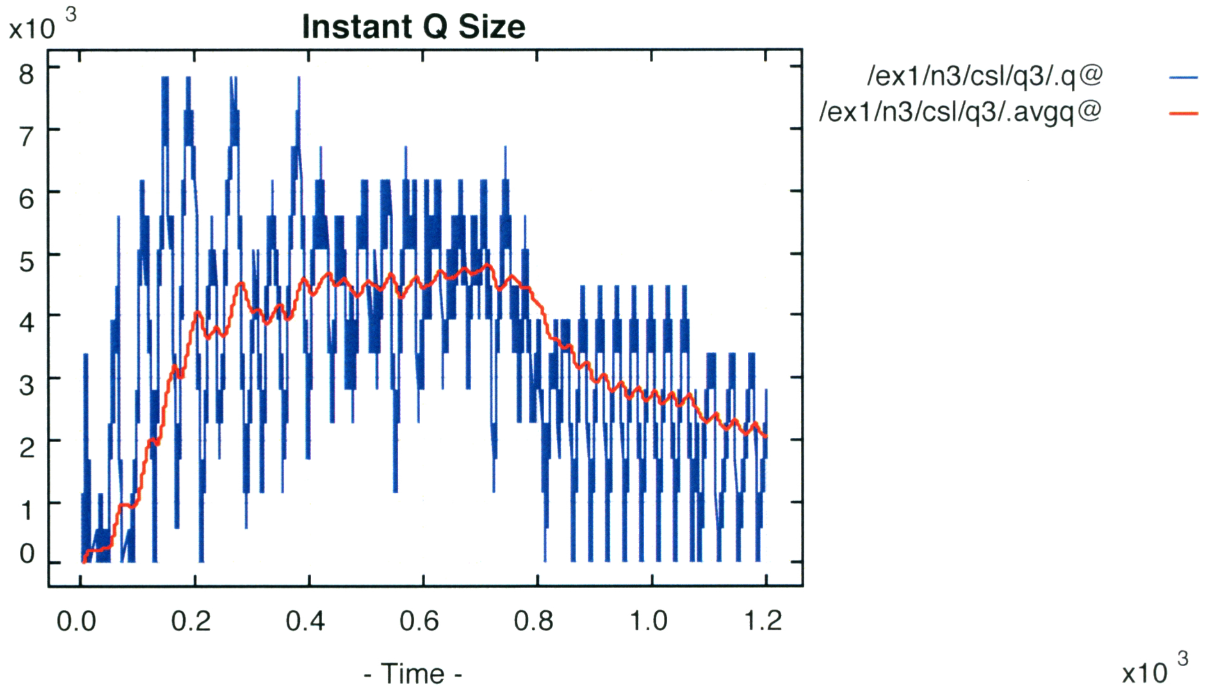
Figure 7.4: Throughput (in bytes per second) vs Time (in second) Plots of (a) Drop-Tail and (b) FAFRED

7.3.5.2 Queue length vs time

Figures 7.5 and 7.6 show the queue length vs time plots of the AQM schemes. For RED, FRED and FAFRED, the y axis is in bytes and for drop-tail the y axis is in the number of packets. The Drop-tail graph is plotted in this way to visualize that it drops all the packets when the buffer is full. In this simulation, the buffer space is for 20 packets. The drop-tail graph shows that it drops all the packets when the queue has 20 packets. In Figures 7.5(a), 7.5(b) and in 7.6(c), the red line is the average queue length and the blue line is the instantaneous queue length. The figures show that FAFRED has the most stable queue length. Its average queue length is the smallest among all schemes.

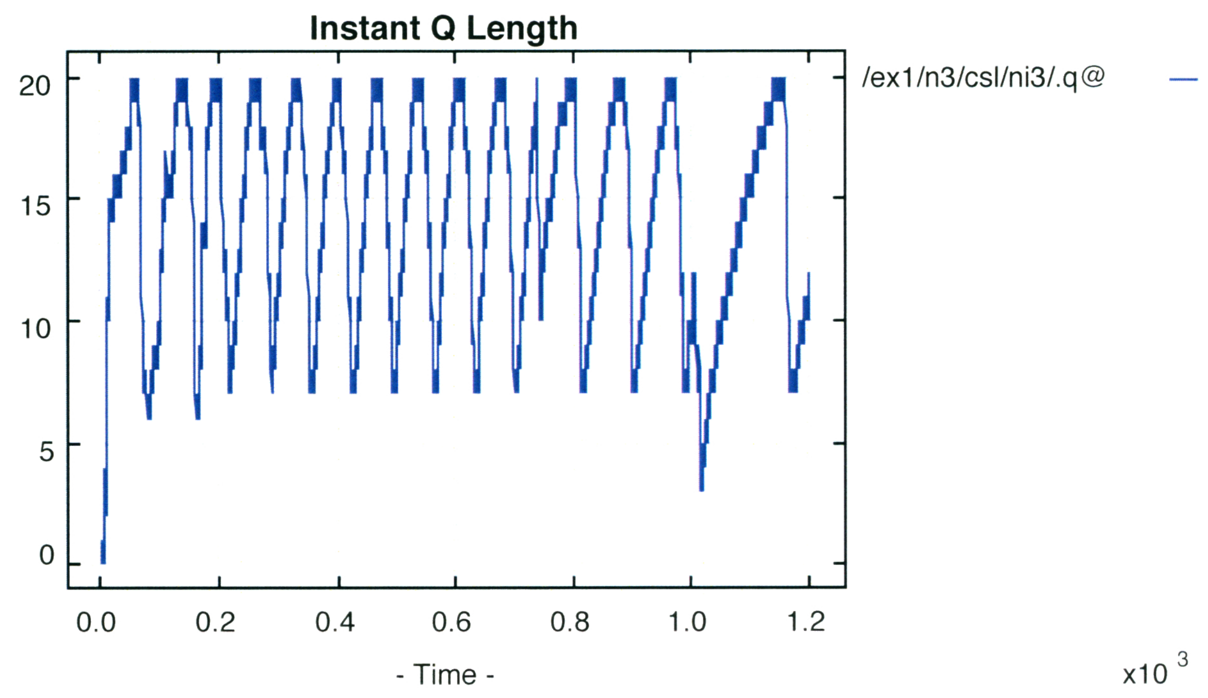


(a)

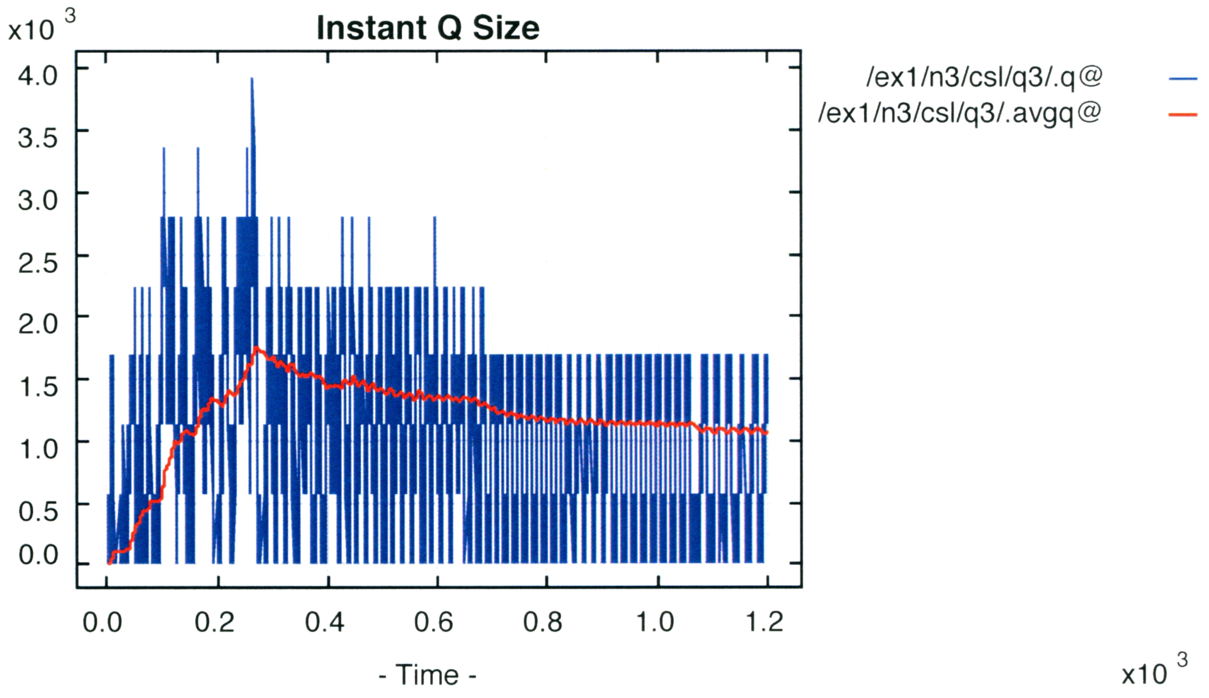


(b)

Figure 7.5: Queue Length (in bytes) vs Time (in second) of (a) RED (b) FRED. The red lines are the average queue length and the blue lines are the instantaneous queue length.



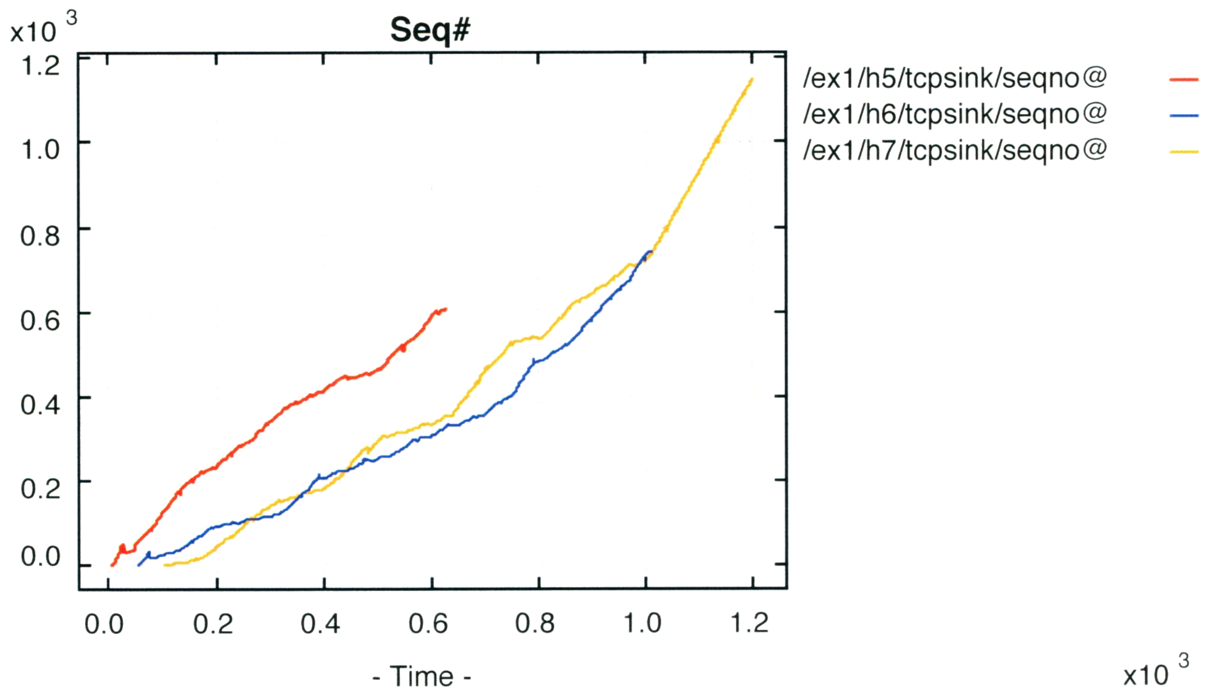
(a)



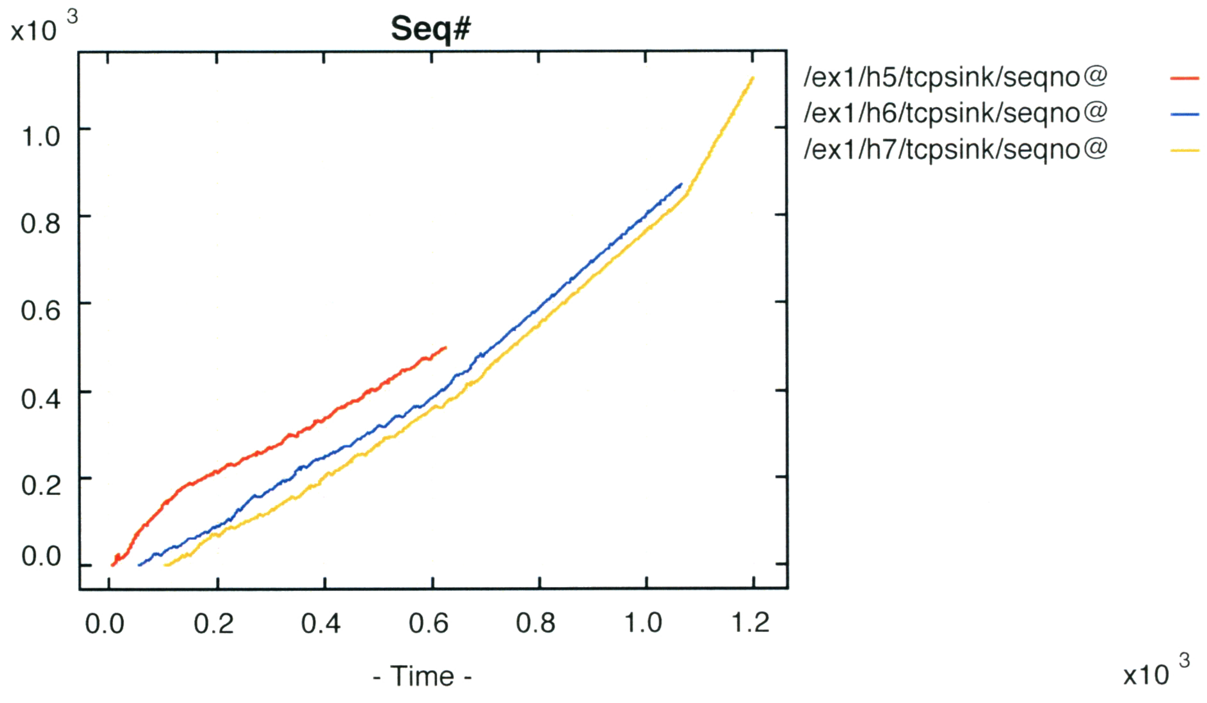
(b)

Figure 7.6: (a) Queue Length (no of packets) vs Time (in second) for Drop-Tail. (b)Queue Length (in bytes) vs Time (in second) of FAFRED. The red line is the average queue length and the blue lines are the instantaneous queue length.

7.3.5.3 TCP Sequence Number vs Time

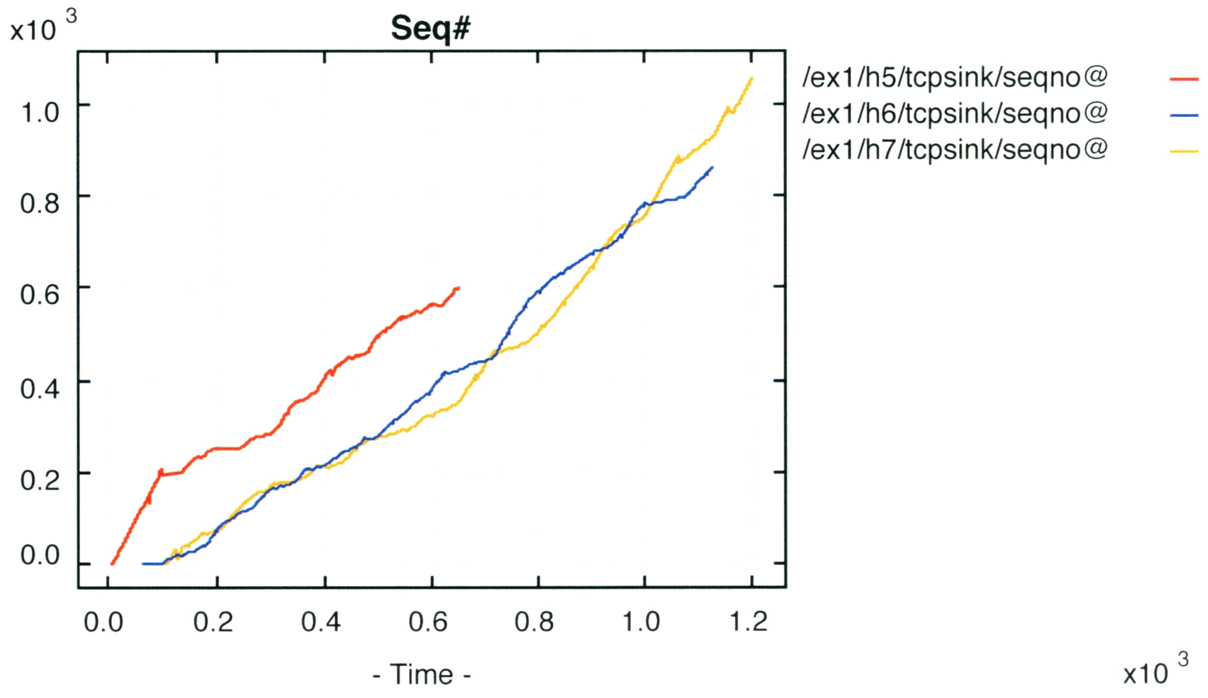


(a)

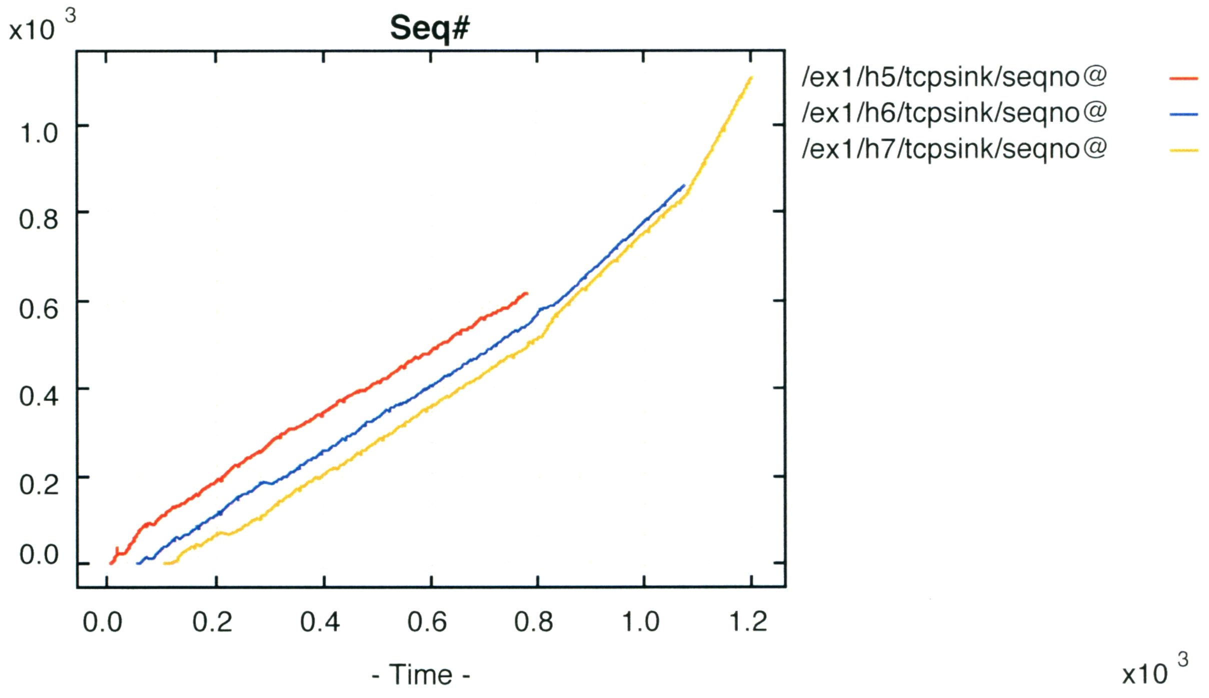


(b)

Figure 7.7: TCP Sequence Number vs Time (in seconds) Plots of (a)RED (b)FRED



(a)



(b)

Figure 7.8: TCP Sequence Number vs Time (in seconds) Plots of (a) Drop-Tail and (b)FAFRED

The TCP sequence number vs time plot reflects another important performance criteria. It provides information about packet dropping and retransmission. If congestion occurs and a packet is dropped, another packet with the same sequence number as the dropped one is retransmitted. That is, the sequence number does not increase with time. In that case, the sequence number vs time graph has a small horizontal section. For optimum performance under TCP, this graph should not have any horizontal sections. Figures 7.7 and 7.8 show the sequence number vs time graphs of the AQM schemes. Drop-tail has many horizontal sections, indicating a large number of retransmissions. The performance of RED is also poor. FAFRED has the least number of horizontal sections among all the studied AQMs.

7.3.5.4 Fairness

Table 7.3: Fairness Index of Various Queue Management Schemes.

RED	Drop-Tail	FRED	FAFRED
0.80	0.66	0.93	0.94

The last performance index studied is fairness. The fairness index is computed using equation (3.3). Ideally we want the fairness index to be close to 1 as a totally fair system has a fairness index of 1. In our simulation, the fairness index is computed based on the individual throughput of the flows when all the three flows are active ($t = 100$ to $t = 600$).

Table 7.3 shows the fairness indices of all the studied queue management schemes. As expected drop-tail has a small fairness index. This is because, when the buffer is full, it drops all the incoming packets. It can happen that all the packets from one flow are dropped. The fairness index of RED is higher than drop-tail. FRED is fairer than RED. FAFRED also has a high fairness index.

7.4 Conclusion

In this chapter the developed fuzzy logic based traffic models were used to build FAFRED, a fair AQM scheme. Several performance indices of FAFRED were compared to other AQM schemes. The results showed promise. By using a genetic algorithm, the FAFRED makes itself adaptive. It maintains a high throughput keeping its queue small and without dropping a large number of packets.

FAFRED does not use any complex mathematical computation and hence can be implemented within the framework of a router without significantly increasing the router load.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

Our work has two major aspects. Firstly, a fuzzy logic based traffic modelling and analysis methodology was developed. This included two traffic models, fuzzy group state model and fuzzy group model (Chapter 5). In the fuzzy group model, a traffic parameter, R, is defined which can be used in traffic analysis and queuing performance prediction. Secondly, we proposed and developed Fuzzy Adaptive Fair Random Early Detection (FAFRED), a router queue management scheme based on our proposed traffic models (Chapter 7). FAFRED is found to be a promising Active Queue Management (AQM) scheme which provides a fair service and performs well.

The following indicates how the research objectives set out in Chapter 1 are addressed. Along with the practical significance of the R parameter and FAFRED, this research makes the following contributions to the field of traffic engineering.

Fuzzy Logic Based Traffic modelling. Existing statistical traffic models rely heavily on complex mathematical computation and extensive measured data. As the speed of networks increases, gathering the necessary quantity of data is becoming more and more difficult. This difficulty may indicate that it is not appropriate to model current or future network traffic using current statistical models. Fuzzy logic based models are capable of modelling a system characterized by imprecise data and require considerably less computation than statistical models. Hence this research proposes a new approach - fuzzy logic based traffic modelling and analysis techniques (objective 2).

The R parameter that is proposed in the fuzzy group model is simple to calculate. Furthermore, R can be estimated off-line for analysis of previously collected traffic traces or on-line for a live traffic stream. For traffic analysis of real traces (objective 5), R was computed for off-line traces using a Perl script. An example is given in Appendix E. In FAFRED, we computed R in real time for (simulated) live traffic flows. The R parameter is estimated using a simple defuzzification method (equation (4.3)). In theory, R can be computed for an off-line trace or a live stream with any number of T-groups. That is, unlike the H parameter, estimation of R does not require a huge data set (objectives 1 and 5).

Indicator of Queuing Performance. A queuing performance indicator is an invaluable tool for network administrators for capacity planning and designing a network. The Hurst parameter which is a property only of self-similar traffic is believed to be an indicator of queuing performance. But as network traffic has recently shown hints of non-stationarity and hence non self-similarity, the Hurst parameter may not be appropriate for all networks. Experimental results show that the H parameter by itself might not be an accurate indicator of queuing performance (Section 6.3). On the other hand, the R parameter, which does not rely on self-similarity or any other assumption appears to be useful as a queuing performance indicator for any network (Section 6.3). Additionally, the R parameter is relatively easy to compute and does not require a huge data set (objectives 1 and 5).

Measure of Burstiness. Theoretically, the H parameter can be a measure of burstiness (Section 2.10) but only under the strict condition that the traffic is self-similar and originates from a high level of aggregation of users or sources. But in modern networks, a burst can arise from even a single user application. Also, experimental results show that a contradictory relationship exists between the H parameter and burstiness (Section 6.4). On the other hand, the R parameter appears to be a reasonably accurate mathematical measure of burstiness of a traffic stream (Section 6.4). The R parameter is based on fuzzy logic and can adapt to diverse traffic conditions (objective 3).

The second part of this research focused on router performance optimization. The idea of using genetic algorithms in the field of network performance optimization is introduced. In particular, this research has made the following contributions to the field of network management.

Representation of a network optimization problem as a Multi-Objective Optimization (MO) Problem. Router performance optimization is a major network optimization problem. In this work the router performance optimization problem is formulated as a multi-objective optimization problem. This lead us to apply multi-objective evolutionary algorithms (MOEA) to router performance optimization. This approach appears to be successful and the author of this research believes that MOEA is a useful approach for solving other network optimization problems with multiple objectives (objective 4).

FAFRED - A Fair and Adaptive Queue Management Scheme. Experiments using FAFRED, the scheme that is developed in this work, show that FAFRED appears to maintain a small router queue and avoid congestion without sacrificing overall throughput. But the main strength of FAFRED is its adaptability. By using the aggregating approach to construct the objective function, multiple performance objectives may be achieved by adjusting the weights in equation (7.2). This approach makes FAFRED more flexible than other AQMs.

In the simulation of FAFRED (Section 7.3), a genetic algorithm is used to find the optimum value of ϵ, the FAFRED input parameter. This approach can also be used to find optimum parameters for other AQM schemes (such as p_{max} in RED). In general, a complex nonlinear relationship exists between the input parameter(s) of an AQM scheme and the router performance objective. A GA approach can be particularly suitable for such cases (objective 4).

8.2 Future Research

This research proposed fuzzy logic based traffic modelling techniques. Traffic models have a wide range of application in the vast area of computer networks. Unfortunately this work was able to cover only a small fraction of such applications.

There are two major parameters that some traffic models generate: packet size distribution and packet inter-arrival distribution. Our proposed fuzzy logic based models do not yet generate either. Generating the packet size distribution is relatively simple. Existing models of packet sizes have been shown to be valid and accurate. Generating packet inter-arrival distribution is much more difficult.

How our fuzzy logic based models can be extended to generate packet inter-arrival distributions should be pursued further in a separate research effort.

Traffic synthesis is an important requirement for any network simulation. A traffic model should be able to synthesize traffic that resembles real network streams. At this point, our proposed models are not capable of traffic synthesis. Future research should explore how a traffic stream can be synthesized for a given value of R.

The use of the fuzzy group model in buffer dimensioning deserves a separate research effort. The group model is developed in a way that the value of R is dependent on buffer capacity. Proper sizing of the router buffer of a network could be computed from the R value of the incoming traffic. Research should be conducted to find a relationship between R and the optimum buffer size.

In developing FAFRED, an aggregating approach is used to find its optimum input parameter. Although an aggregating approach is the simplest way to deal with an Multi-objective Optimization problem, it might not be the most accurate for all situations. For a network with complex and heterogeneous traffic, the aggregating approach might not provide optimum performance. The use of other MOEA techniques, such as Vector Evaluated Genetic Algorithms (Section 4.2.3.1) to determine the optimum input parameter(s) for FAFRED is a major topic for future research.

During our FAFRED experiments, a feature of J-Sim was used to add attacks to the input traffic stream. It was noticed that the value of R changed immediately. Further exploration may indicate that another valuable use of R is as an on-line detector or perhaps indicator of network attacks.

The validation of a FL system requires stability analysis and robustness analysis with some analytic tool. In FAFRED, the network is trained to get optimum values for the input parameters. In other words inference rules (7.1) are dynamic with some feedback from the previous state. That is the inference rule is dynamic and dependent on the previous states. Existing analytic tools such as FL toolbox [68]that are used to study the validation a FL system are not suitable for such a dynamic system, because FL toolbox requires a static matrix for the inferencing rulebase. Future research should explore to develop an analytic tool that can study the stability and robustness of a dynamic FL system like FAFRED.

References

[1] J. Abate, Choudhury G.L., and W. Whitt. Asymptotics for steady-state tail probabilities in structured markov queueing models. *Stochastic Models*, 10(1):99–143, 1994.

[2] J. Abbate. *From ARPANET to Internet: a history of ARPA-sponsored computer networks, 1966-1988.* PhD dissertation, University of Pennsylvania, United States – Pennsylvania, 1994.

[3] P. Abry and D. Veitch. Wavelet analysis of long-range-dependent traffic. *IEEE Transactions on Information Theory*, 44(1):2–15, 1998.

[4] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers.

[5] Å. Arvidsson, M. Roughan, and T. Rydén. On the origins of long-range dependence in TCP traffic. in ITC-17, 2001.

[6] K. Avrachenkov, U. Ayesta, E. Altman, Nain, P., and C. Barakat. The effect of router buffer size on the TCP performance. In *Proceedings of the LONIIS Workshop on Telecommunication Networks and Teletraffic Theory*, pages 116–121, 2002.

[7] T. Back, D.B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation.* IOP Publishing Ltd, Bristol, UK, 1997.

[8] D. Barman, G. Smaragdakis, and I. Matta. The Effect of Router Buffer Size on HighSpeed TCP Performance. In *IEEE Globecom 2004 - Global Internet and Next Generation Networks*, Dallas, TX, December 2004.

[9] P.P. Bonissone, V. Badami, K.H. Chiang, P.S. Khedkar, K.W. Marcelle, and M.J. Schutten. Industrial applications of fuzzy logic at General Electric. *Proceedings of the IEEE*, 83(3):450–465, 1995.

[10] N. Brownlee. Traffic Flow Measurement: Experiences with NeTraMet. Internet Rfcs, RFC2123, March 1997.

[11] R. Càceres, P.B. Danzig, S. Jamin, and D.J. Mitzel. Characteristics of wide-area TCP/IP conversations. *Proceedings of the Conference on Communications Architecture & Protocols*, pages 101–112, 1991.

[12] P. Calyam and C.G. Lee. Characterizing voice and video traffic behavior over the Internet. *International Symposium on Computer and Information Sciences (ISCIS)*, 2005.

[13] P. Calyam, M. Sridharan, W. Mandrawa, and P. Schopis. *Passive and Active Measurement*, chapter Performance measurement and analysis of H. 323 Traffic, pages 137–146. Springer, NY, 2004.

[14] J. Cao, WS Cleveland, Y. Gao, K. Jeffay, FD Smith, and M. Weigle. Stochastic models for generating synthetic HTTP source traffic. *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, pages 1547–1558, March 2004.

[15] J. Cao, W.S. Cleveland, D. Lin, and D.X. Sun. On the nonstationarity of Internet traffic. In *Proceedings of the 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 102–112. ACM Press New York, NY, USA, 2001.

[16] J. Cao, W.S. Cleveland, D. Lin, and D.X. Sun. Internet traffic tends toward Poisson and independent as the load increases. *Nonlinear Estimation and Classification*, pages 83–109, 2002.

[17] J. Cao, WS Cleveland, and DX Sun. Fractional Sum-Difference Models for Open-Loop Generation of Internet Packet Traffic. Technical report, Bell Labs, Murray Hill, NJ, 2002.

[18] T.S. Choi, C.H. Kim, S. Yoon, J.S. Park, B.J. Lee, H.H. Kim, H.S. Chung, and T.S. Jeong. Content-aware Internet application traffic measurement and analysis. In *Network Operations and Management Symposium, 2004. IEEE/IFIP*, volume 1, pages 511–524, April 2004.

[19] M.Y. Chow and H. Tram. Application of fuzzy logic technology for spatial load forecasting. *IEEE Transactions on Power Systems*, 12(3):1360–1366, 1997.

[20] C.K. Chui. *An introduction to wavelets.* Academic Press, New York, January 1992.

[21] J. Chung and M. Claypool. Analysis of active queue management. In *Second IEEE International Symposium on Network Computing and Applications*, pages 359–366, April 2003.

[22] K.C. Claffy, H.W. Braun, G.C. Polyzos, S. Center, G. Atomics, and C.A. San Diego. A parameterizable methodology for Internet traffic flow profiling. *IEEE Journal on Selected Areas in Communications*, 13(8):1481–1494, 1995.

[23] B. Claise. Cisco systems NetFlow services export. Technical Report 9, IETF Internet Draft, August, 2002.

[24] M. Claypool, R. Kinicki, M. Li, J. Nichols, and H. Wu. Inferring Queue Sizes in Access Networks by Active Measurement. *Proceedings of the 5th Passive and Active Measurement Workshop (PAM)*, pages 227–236, 2004.

[25] R.G. Clegg. A Practical Guide to Measuring the Hurst Parameter. *International Journal of Simulation: Systems, Science & Technology*, pages 3–14, 2006.

[26] C.A.C. Coello, D.A. Van Veldhuizen, and G.B. Lamont. *Evolutionary algorithms for Solving Multi-objective Problems.* Kluwer Academic, Berlin, Germany, 2002.

[27] J.L. Cohon. *Multiobjective Programming and Planning.* Dover Publications, NY, 2004.

[28] J.L. Cohon and D.H. Markes. A Review and Evaluation Of Multiobjective Programming Techniques. *Water Resources Research*, pages 208–220, April 1975.

[29] G. Combs. Ethereal: A network protocol analyzer. Computer Software, Available at http://www.ethereal.com, 2004.

[30] D. Comer. *Internetworking with TCP/IP*, volume 1. Prentice-Hall, Upper Saddle River, NJ, 1991.

[31] D.W. Corne, J.D. Knowles, and M.J. Oates. The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 839–848, 2000.

[32] M.E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[33] I. Daubechies. *Ten Lectures on Wavelets.* Society for Industrial & Applied Mathematics, Philadelphia, USA, 1992.

[34] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley and Sons, NJ, USA, 2001.

[35] A. Dhamdhere and C. Dovrolis. Open issues in router buffer sizing. *ACM SIGCOMM Computer Communication Review*, 36(1):87–92, 2006.

[36] A. Dhamdhere, H. Jiang, and C. Dovrolis. Buffer Sizing for Congested Internet Links. In *24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2005*, pages 1072–1083, March 2005.

[37] P. Doukhan, G. Oppenheim, and M.S. Taqqu. *Theory and Applications of Long-range Dependence.* Birkhäuser, Boston, USA, 2003.

[38] NG Duffield and N. OConnell. Large deviations and overflow probabilities for the general single-server queue, with applications. *Mathematical Proceedings of the Cambridge Philosophical Society*, 118(2):363–374, 1995.

[39] A. Erramilli, M. Roughan, D. Veitch, and W. Willinger. Self-similar traffic and network dynamics. *Proceedings of the IEEE*, 90(5):800–819, 2002.

[40] A. Erramilli, RP Singh, and P. Pruthi. Chaotic maps as models of packet traffic. In *Proceedings of 14th International Teletraffic Congress*, volume 1, pages 329–338, Juan-les-Pins, France, June 1994.

[41] D. Estrin and DJ Mitzel. An assessment of state and lookup overhead in routers. In *Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 2332–2342, 1992.

[42] J. Fan and Q. Yao. *Nonlinear Time Series: Nonparametric and Parametric Methods.* Springer, New York, 2003.

[43] W. Feller. *An Introduction to Probability Theory and Its Applications, Vol. 1.* John Wiley & Sons, New York, 1957.

[44] W. Fischer and K. Meier-Hellstern. The Markov-modulated Poisson process (MMPP) cookbook. *Performance Evaluation*, 18(2):149–171, 1993.

[45] S. Floyd. RED: Discussions of setting parameters. World Wide Web Document, Available at: http://www.aciri.org/floyd/REDparameters.txt, November 1997.

[46] S. Floyd. Recommendation on using the gentle variant of RED. note, available at http://www. icir. org/floyd/red/gentle. html, March, 2000.

[47] S. Floyd and V. Jacobson. Random Early Detection (RED) gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[48] C.M. Fonseca and P.J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[49] VS Frost and B. Melamed. Traffic modeling for telecommunications networks. *Communications Magazine, IEEE*, 32(3):70–81, 1994.

[50] K. Fukuda, M. Takayasu, and H. Takayasu. A cause of self-similarity in TCP traffic. *International Journal of Communication Systems*, 18(6):603, 2005.

[51] A.C. Gilbert. Multiscale analysis and data networks. *Applied and Computational Harmonic Analysis*, 10(3):185–202, 2001.

[52] J. Gleick. *Chaos.* Viking, New York, 1987.

[53] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[54] M. Grossglauser and J.C. Bolot. On the relevance of long-range dependence in network traffic. *IEEE/ACM Transactions on Networking (TON)*, 7(5):629–640, 1999.

[55] P. Hajela and C.Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural and Multidisciplinary Optimization*, 4(2):99–107, 1992.

[56] S.H. Han, M.S. Kim, H.T. Ju, and J.W.K. Hong. The Architecture of NG-MON: A Passive Network Monitoring System for High-Speed IP Networks. *Proceeding of 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, pages 16–27, 2002.

[57] M. Hassan and R. Jain. *High Performance TCP/IP Networking: Concepts, Issues, and Solutions.* Prentice-Hall, Upper Saddle River, NJ, 2004.

[58] H. Heffes and D. Lucantoni. A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance. *IEEE Journal on Selected Areas in Communications*, 4(6):856–868, 1986.

[59] J. Heitkotter and D. Beasley. The Hitch-Hiker's Guide to Evolutionary Computation: A list of Frequently Asked Questions (FAQ). USENET: comp.ai.genetic, Available via anonymous FTP from rtmf.mit.edu/pub/usenet/news.answers/aifaq/genetic/, 1996.

[60] L.P. Holmblad and J.J. Ostergaard. Control of a cement klin by fuzzy logic techniques. In *Proceedings of the 8th IFAC Conference*, pages 809–814, Kyoto, Japan, 1981.

[61] V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM Computer Communication Review*, 25(1):157–187, 1995.

[62] V. Jacobson, C. Leres, S. McCanne, et al. Tcpdump. available via anonymous FTP to ftp.ee.lbl.gov, 1989.

[63] S. Jaffard, Y. Meyer, and R.D. Ryan. *Wavelets: tools for science & technology.* Society for Industrial & Applied Mathematics, Philadelphia, USA, 2001.

[64] D.L. Jagerman and B. Melamed. Burstiness Descriptors of Traffic Streams: Indices of Dispersion and PeakednessVoice over IP and jitter avoidance on low speed links. In *Proceedings of the Twenty Eighth Annual Conference on Information Sciences and Systems*, volume 1, pages 1–5, 1994.

[65] D.L. Jagerman, B. Melamed, and W. Willinger. Stochastic modeling of traffic processes. *Crc Probability And Stochastics Series*, pages 271–320, 1998.

[66] R. Jain, A. Durresi, and G. Babic. Throughput Fairness Index: An Explanation. ATM Forum/99-0045. http://www.cse.wustl.edu/jain/atmf/a99-0045.htm, 1999.

[67] R. Jain and S. Routhier. Packet Trains–measurements and a new model for computer network traffic. *IEEE Journal on Selected Areas in Communications*, 4(6):986–995, 1986.

[68] J.S.R. Jang and N. Gulley. *MATLAB Fuzzy Logic Toolbox User Guide (Version 1)*. The MathWorks, Inc, 1997.

[69] HD Jeong. *Modelling of Self-similar Teletraffic for Simulation.* PhD dissertation, Department of Computer Science, University of Canterbury, NZ, 2002.

[70] H. Jiang and C. Dovrolis. Why is the internet traffic bursty in short time scales? In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 241–252. ACM Press New York, NY, USA, 2005.

[71] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido. A nonstationary Poisson view of Internet traffic. In *Proceedings of the Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1558–1569, March 2004.

[72] S. Keshav and R. Sharma. Issues and trends in router design. *Communications Magazine, IEEE*, 36(5):144–151, 1998.

[73] H. Kettani and J.A. Gubner. A novel approach to the estimation of the Hurst parameter in self-similar traffic. In *Proceedings of the 27th Annual IEEE International Conference on Local Computer Networks*, pages 160–165, November 2002.

[74] R.R. Kompella and C. Estan. The power of slicing in internet flow measurement. *Proceedings of the Internet Measurement Conference 2005 on Internet Measurement Conference table of contents*, pages 9–9, 2005.

[75] Y.J. Lai and C.L. Hwang. *Fuzzy Multiple Objective Decision Making: Methods and Applications.* Springer-Verlag, New York, USA, 1994.

[76] D. Le Gall. MPEG: a video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, 1991.

[77] CC Lee. Fuzzy logic in control systems: Fuzzy logic controller–part I. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):404–418, 1990.

[78] KC Lee and P. Gardner. Adaptive Neuro-Fuzzy Inference System (ANFIS) Digital Predistorter for RF Power Amplifier Linearization. *Vehicular Technology, IEEE Transactions on*, 55(1):43–51, 2006.

[79] WE Leland, MS Taqqu, W. Willinger, and DV Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE Transactions on Networking*, 2(1):1–15, 1994.

[80] D. Lin and R. Morris. Dynamics of random early detection. In *Proceedings of the ACM SIGCOMM'97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 127–137, New York, USA, 1997. ACM Press.

[81] D.V. Lindley. The theory of queues with a single server. *Proceedings of the Cambridge Philosophical Society*, 48:277–289, 1952.

[82] R.J.A. Little and D.B. Rubin. *Statistical analysis with missing data.* John Wiley & Sons, Inc., New York, USA, 1986.

[83] J.W. Lockwood, C. Neely, C. Zuver, J. Moscola, S. Dharmapurikar, and D. Lim. An extensible, system-on-programmable-chip, content-aware Internet firewall. In *Proceedings of Field Programmable Logic and Applications*, pages 859–868, New York, USA, 2003. Springer.

[84] S.H. Low, F. Paganini, J. Wang, S. Adlakha, and J.C. Doyle. Dynamics of TCP/RED and a scalable control. In *Proceedings of IEEE INFOCOM*, pages 239–248, 2002.

[85] R.D. Maher III, V.A. Bennett, A.V. Rana, M.A. Lie, K.W. Brandon, M.W. Hervin, and C.A. Garrow. Content aware network apparatus, November 25 2003. US Patent 6,654,373.

[86] SC Malik and S. Arora. *Mathematical Analysis.* John Wiley & Sons Inc, New York, USA, 1992.

[87] G. Malkin. Traceroute Using an IP Option. Internet Request for Comments Document, RFC1393, January 1993.

[88] E.H. Mamdani. Application of fuzzy logic to approximate reasoning using linguistic synthesis. In *Proceedings of the sixth International Symposium on Multiple-valued Logic*, pages 196–202, CA, USA, 1976. IEEE Computer Society Press.

[89] BB Mandelbrot. *The fractal geometry of nature.* Freeman, New York, USA, 1983.

[90] R.T. Marler and J.S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[91] W. Matthews. Passive and Active Monitoring on a High Performance Research Network. Technical Report SLAC-PUB-8776, Stanford Linear Accelerator Center, Menlo Park, CA, USA, May 2001.

[92] G. Mayor and J. Silvester. A Trace-Driven Simulation of an ATM Queueing System with Real Network Traffic. *Proc. of IEEE ICCCN*, pages 28–32, 1996.

[93] J. Mendel. Fuzzy Logic systems for engineers: a tutorial. In *Proceedings of the IEEE*, volume 83, pages 345–377, March 1995.

[94] K.M. Miettinen. *Nonlinear Multiobjective Optimization.* Springer, New York, 1999.

[95] J.A. Miller, R. Nair, Z. Zhang, and H. Zhao. JSIM: A Java-Based Simulation and Animation Environment. *Proceedings of the 30th Annual Simulation Symposium*, pages 31–42, 1997.

[96] F.J. Molz, H.H. Liu, and J. Szulga. Fractional Brownian motion and fractional Gaussian noise in subsurface hydrology: A review, presentation of fundamental properties, and extensions. *Water Resources Research*, 33(10):2273–2286, 1997.

[97] G.C. Mouzouris and J.M. Mendel. Non-singleton fuzzy logic systems. In *Proceedings of the Third IEEE Conference on Fuzzy Systems.*, pages 456–461, 1994.

[98] PMA NLANR. Special Traces Archive. World Wide Web Document, Available at: http://pma.nlanr.net/special/index.html, January 2006.

[99] I. Norros. A storage model with self-similar input. *Queueing Systems*, 16(3):387–396, 1994.

[100] H. Ono, T. Ohnishi, and Y. Terada. Combustion control of refuse incineration plant by fuzzy logic. *Fuzzy Sets and Systems*, 32(2):193–206, 1989.

[101] TJ Ott, TV Lakshman, and LH Wong. SRED: stabilized RED. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1346–1355, New York, USA, March 1999.

[102] C. Park, F. Hernández-Campos, JS Marron, and F.D. Smith. Long-range dependence in a changing Internet traffic mix. *Computer Networks*, 48(3):401–422, 2005.

[103] K. Park, G. Kim, and M.E. Crovella. Effect of traffic self-similarity on network performance. *Proceedings of SPIE*, 3231:296, 1997.

[104] V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *Networking, IEEE/ACM Transactions on*, 3(3):226–244, June 1995.

[105] V. Paxson and S. Floyd. Why We Don't Know How To Simulate The Internet. *Simulation Conference Proceedings, 1997. Winter*, pages 1037–1044, 1997.

[106] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. Internet Request For Comments Documents, RFC 3168, 2001.

[107] R. Ritke, X. Hong, and M. Gerla. Contradictory relationship between Hurst parameter and queueing performance (extended version). *Telecommunication Systems*, 16(1):159–175, 2001.

[108] D.A. Rolls, G. Michailidis, and F. Hernández-Campos. Queueing analysis of network traffic: methodology and visualization tools. *Computer Networks*, 48(3):447–473, 2005.

[109] H.J. RONG, N. SUNDARARAJAN, G.B. HUANG, and P. SARATCHANDRAN. Sequential Adaptive Fuzzy Inference System(SAFIS) for nonlinear system identification and prediction. *Fuzzy sets and systems*, 157(9):1260–1275, 2006.

[110] O. Rose. Estimation of the Hurst Parameter of Long-Range Dependent Time Series. Technical Report 137, University of Würzburg, Würzburg, Germany, 1996.

[111] M. Roughan, D. Veitch, and P. Abry. On-line estimation of the parameters of long-range dependence. In *Proceedings of the IEEE Global Telecommunications Conference*, volume 6, pages 3716–3721, Sydney, Australia, November 1998.

[112] J.D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, NJ, USA, 1985. Lawrence Erlbaum Associates.

[113] H. Schwetman. Csim18-the simulation engine. In *Proceedings of the Simulation Conference*, pages 517–521, 1996.

[114] Q. Song, T. Ma, and N. Kasabov. Transductive Knowledge Based Fuzzy Inference System for Personalized Modeling. *Lecture Notes of Artificial Intelligence*, 3614:528–535, 2005.

[115] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[116] R.G.D. Steel and J.H. Torrie. *Principles and Procedures of Statistics.* McGraw-Hill, New York, USA, 1981.

[117] M. Sugeno. *Industrial Applications of Fuzzy Control.* Elsevier Science Inc., New York, USA, 1985.

[118] Cisco Systems. Buffer Tuning for all Cisco Routers. Document ID 15901, World Wide Web Document, Available at: http://www.cisco.com/warp/public/63/buffertuning.html, January 2007.

[119] A.S. Tanenbaum. *Computer Networks.* Prentice-Hall, Upper Saddle River, NJ, fourth edition, 2002.

[120] C. Villamizar and C. Song. High Performance TCP in ANSNET. *ACM SIGCOMM Computer Communication Review*, 24(5):45–60, 1994.

[121] M. Voznak. Voice over IP and jitter avoidance on low speed links. In *Proceedings of the International Conference in Research in Telecommunication Technology*, pages 80–227, Zilina, Slovakia, 2002.

[122] L.X. Wang. *Adaptive Fuzzy Systems and Control: Design and Stability Analysis.* Prentice-Hall, Upper Saddle River, NJ, 1994.

[123] W.X. Wang, B.H. Wang, B. Hu, G. Yan, and Q. Ou. General Dynamics of Topology and Traffic on Weighted Technological Networks. *Physical Review Letters*, 94(18):188702, 2005.

[124] W. Willinger. *The discovery of self-similar traffic*, volume 1769. Springer-Verlag, London, UK, 2000.

[125] W. Willinger and V. Paxson. Where Mathematics meets the Internet. *Notices of the American Mathematical Society*, 45(8):961–970, 1998.

[126] W. Willinger, V. Paxson, and M.S. Taqqu. *Self-similarity and heavy tails: structural modeling of network traffic, a practical guide to heavy tails: statistical techniques for analyzing heavy tailed distributions*, pages 27–53. Birkhauser-Verlag, Boston, 1998.

[127] R.R. Yager and D.P. Filev. *Essentials of fuzzy modeling and control.* Wiley-Interscience, New York, USA, 1994.

[128] O. Yagishita, O. Itoh, and M. Sugeno. Application of fuzzy reasoning to the water purification process. *Industrial Applications of Fuzzy Control*, pages 19–40, 1990.

[129] L.A. Zadeh. Soft computing and fuzzy logic. *Software, IEEE*, 11(6):48–56, 1994.

[130] L.A. Zadeh. Fuzzy logic=computing with words. *IEEE Transactions on Fuzzy Systems*, 4(2):103-111, May 1996.

[131] L.A. Zadeh. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi A Zadeh*, volume 6 of *Advances in Fuzzy Logic.* World Scientific, Singapore, May 1996.

[132] Lofti A. Zadeh. Fuzzy logic. *Computer*, 21(4):83–93, 1988.

[133] L. Zhao, Y.C. Lai, K. Park, and N. Ye. Onset of traffic congestion in complex networks. *Physical Review E*, 71(2):26125, 2005.

[134] E. Zitzler. *Evolutionary algorithms for multiobjective optimization.* PhD dissertation, Swiss Federal Institute of Technology, Zurich, Switzerland, November 1999.

[135] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

Appendix A

Glossary

AQM	Active Queue Management. A technique of preventing congestion in packet-switched networks.
bandwidth	The maximum speed at which data can be transferred between the sender and receiver once transmission has begun.
burst-within-burst	A burst within another burst. This phenomenon is observed in self-similar traffic.
channel capacity	same as bandwidth.
chromosome	An data structure by which an individual is represented. Generally this is a string of bits {0,1}.
circuit-switched network	A network where a dedicated channel (circuit) is established for the entire duration of transmission of a conversation.
communication media	The actual physical (or wireless) path over which the electrical or optical signal travels. The most common communication media are twisted-pair cable, coaxial cable and fiber optic cable.
conditional probability	The conditional probability, denoted by $P(B\|A)$, is the probability of B occurring given that A has occurred. The conditional probability $P(B\|A)$ is defined as, $$P(B\|A) = \frac{P(A,B)}{P(A)}$$

congestion	The situation, when incoming traffic is so heavy that router buffer overflow occurs.
crossover	A genetic operator where two chromosomes exchange their genes.
datagram	An IP layer PDU.
defuzzification	The mechanism for converting the output fuzzy sets of a inference engine into nonfuzzy or crisp values.
delay	The amount of time taken by a packet to be transmitted from source to destination.
encapsulation	The process of adding headers to the user data in each layer in a layered protocol system.
filter(s)	A set of rules that define a firewall.
firewall	A hardware or software that prevents unauthorized access to network resources.
fuzzification	The method of assigning linguistic values to a variable using (a relatively small number of) membership functions.
Gaussian White Noise	A white noise that has a normal distribution.
gene	A bit position in a chromosome.
goodput	The amount of this useful information that is delivered per second to the application layer protocol
jitter	Unwanted abrupt variation of of one or more signal characteristics such as delay. Jitter that is caused by packets arriving via different routes is called extreme jitter.
latency	Time required to transfer an empty message from sender to receiver.
Markov chain	A stochastic process with the Markov property.
Markov property	A property where the future state of a process is conditionally independent of the past states.
membership function	A graphical (or tabulated numeric) function that assigns a membership values between 0 and 1 to the crisp values of an input variable over its universe of discourse.

mutation	A process of flipping a randomly selected bit in the offspring.
open loop traffic	A congestion-free traffic.
over-provisioned link	A link that can not be congested.
packet-switched network	A network where messages are divided into packets which are sent individually.
protocol stack	A layered set of protocols that perform the entire communication task. The layers works together and each layer performs a predefined set of functions.
protocol data unit (PDU)	A unit of data that is specified in a protocol of a given layer in a layered protocol stack. A PDU consists of control information of the given layer and possibly user data of that layer.
random process	A time-indexed series of random variables.
router	A device that determines proper path for a packet to travel between different networks.
routing	A process that determines the proper path for a packet to travel between different networks.
segment	A TCP layer PDU.
server	An application program (or a computer running that application) that accepts connections in order to service requests by sending back responses.
sojourn time	Time spent in a state.
stochastic process	A time-indexed series of random variables.
throughput	The amount of digital data per unit time that is delivered to a certain host in a network.
white noise	A random process with a flat power spectral density.
wrapper	A software that contains other software or data, so that the content elements are disguised or hidden.

Appendix B

List of Abbreviations

AI	Artificial Intelligence
ANFIS	Adaptive Network based Fuzzy Inference System
AQM	Active Queue Management
ARIMA	Auto-Regressive Integrated Moving Average
BDP	Bandwidth Delay Product
BSCL	Buffer Sizing of Congested Link
CCDF	Complementary Cumulative Distribution Function
CP	Congestion Prediction
DM	Decision Maker
EA	Evolutionary Algorithm
EC	Evolutionary Computing
ECN	Explicit Congestion Notification
FAFRED	Fuzzy Adaptive Fair Random Early Detection
FARIMA	Fractional Auto-Regressive Integrated Moving Average
fBm	fractional Brownian motion
fGn	fractional Gaussian noise
FL	Fuzzy Logic
FRED	Fair Random Early Detection
FSD	Fractional Sum Difference
FSD-MA	Fractional Sum Difference - Moving Average
FTP	File Transfer Protocol
GA	Genetic Algorithm
Gbps	Giga bit per second

HTTP	Hyper Text Transfer Protocol
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
i.i.d.	independent and identically distributed
LAN	Local Area Network
LRD	Long Range Dependent
MA	Moving Average
MO	Multi-objective Optimization
MOEA	Multi-objective Optimization Evolutionary Algorithm
MMPP	Markov Modulated Poisson Process
MRA	Multi-Resolution Analysis
NAT	Network Address Translation
PD	Packet Drop
PQM	Passive Queue Management
QM	Queue Manager
QoS	Quality of Service
RED	Random Early Detection
RTT	Round Trip Time
SAFIS	Sequential Adaptive Fuzzy Inference System
SC	Soft Computing
SIC	Sensitive-dependence on Initial Condition
SRD	Short Range Dependence
SRED	Stabilized random Early Detection
SS	Self Similar
TCP	Transport Control Protocol
UDP	User Datagram Protocol
VEGA	Vector Evaluated Genetic Algorithm
VOIP	Voice Over Internet Protocol
WSS	Weak-Sense Stationary
WWW	World Wide Web

Appendix C

An example of a Genetic Algorithm

Genetic algorithms operate on a initial population of potential solutions to an optimization problem and apply the principles of survival of the fittest to produce better candidates as a solution. A possible or candidate solution is called an *individual.* An individual is represented by a data structure called a *chromosome.* Generally this is simply a string of bits $\{0,1\}$. Each bit position in a chromosome (bit string) is called a *gene.*

The starting point of a GA is to generate an *initial* population. The initial population generally contains random bit strings. Each member of this initial population is evaluated for its *fitness.* The fitness value indicates how strong an individual is as a solution. The genetic operators (reproduction, crossover and mutation) then operate on the initial population of chromosomes to generate a new population. These operators are modelled on their biological counterparts. Reproduction is a *selection* process where individuals are chosen for the next generation with selection probability proportional to the fitness value. Crossover is a method where pairs of chromosomes (parents) in the new population are selected at random to exchange their genetic material (bits). Crossover produces two new chromosomes. A process of flipping a randomly selected bit in the offspring is called *mutation.* Thus with the help of these operators, a new population is generated and the old population is replaced by the new one. At this stage the GA has performed one *generation.* This process is repeated and the population evolves and another generation is produced. The new generation contains more and more highly fit chromosomes. This evolution continues until the convergence criterion is reached.

As an example let us consider a very simple problem of the sum of digits of a bit chromosome of length 10. Let the following be the GA parameters:

Population Size	=	4
Crossover Rate	=	0.7
Number of Crossover Point	=	1
Mutation Rate	=	0.1
Maximum Number of Generation	=	5

The first parameter Population Size indicates that the initial population consists of 4 chromosomes. Crossover Rate indicates whether a chromosome will be picked for crossover or not. For each chromosome, a random number between 0 and 1 is generated. If this generated number is less than the crossover rate (0.7), the chromosome will be selected for crossover. The third parameter indicates that one point crossover will be used.

The GA starts with initial population. Let the randomly generated population be

$$\begin{aligned} P0 &: \text{0000110101 fitness} = 4 \\ P1 &: \text{1100010100 fitness} = 4 \\ P2 &: \text{1111001000 fitness} = 5 \\ P3 &: \text{0101010110 fitness} = 5 \end{aligned} \tag{C.1}$$

Once we have the initial population, genetic operators are applied to get new population.

Selection

The first genetic operator that can be applied is selection. A popular selection algorithm is the *roulette wheel* method. In this method the cumulative fitness (cfitness) of each chromosome is computed

$$\begin{aligned} P0 &: \text{0000110101, cfitness} = 0.2222 \\ P1 &: \text{1100010100, cfitness} = 0.4444 \\ P2 &: \text{1111001000, cfitness} = 0.7222 \\ P3 &: \text{0101010110, cfitness} = 1.0000 \end{aligned} \tag{C.2}$$

For $P0$, fitness is 4, hence its cfitness is 4/(4+4+5+5)=0.2222. For $P2$, fitness is 4 and the cfitness is (fitness of $P0$)+$\frac{4}{4+4+5+5}$, which equates to 0.4444. Similarly to $P2$, cfitness of the other members of the population are computed. Then four (since population size is 4) random numbers between 0 and 1 are generated to determine which chromosome gets selected. The cumulative fitness values that bracket the random number determine the chromosome to be selected. For example according to equation (C.2), if the generated random number is between 0 to 0.2222, chromosome $P0$ is selected. A possible selection procedure for 4 randomly generated numbers ($R0$ to $R3$) is shown in equation (C.3) and the new population is shown in equation (C.4).

$$\begin{aligned} R0 &: 0.0752,\ \text{selected} = P0 \\ R1 &: 0.5479,\ \text{selected} = P2 \\ R2 &: 0.9502,\ \text{selected} = P3 \\ R3 &: 0.6546,\ \text{selected} = P2 \end{aligned} \tag{C.3}$$

$$\begin{aligned} P0 &: 0000110101\ \text{fitness} = 4 \\ P1 &: 1111001000\ \text{fitness} = 5 \\ P2 &: 0101010110\ \text{fitness} = 5 \\ P3 &: 1111001000\ \text{fitness} = 5 \end{aligned} \tag{C.4}$$

Crossover

In crossover two chromosomes exchange their genes. The crossover probability determines which chromosome undergoes crossover. For each chromosome a random number between 0 and 1 is generated. If the number is less than the crossover probability, the chromosome gets picked. Each time a pair of chromosomes are picked for crossover. A crossover point on the parent chromosome is selected. In a *one point crossover*, all genes beyond that point are swapped between the two parent chromosomes. In a *two point crossover*, two points are selected on parent chromosomes. All the genes between the two points are swapped. A possible crossover between chromosome $P2$ and $P3$ with crossover point 7 is shown in equation(C.5) and the new population is shown in equation(C.6).

$$\begin{aligned} 0101010\,|\,110 \quad &\text{is replaced by} \quad 0101010\,|\,000 \\ 1111001\,|\,000 \quad &\text{is replaced by} \quad 1111001\,|\,110 \end{aligned} \tag{C.5}$$

$$\begin{aligned} P0 &: 0000110101 \text{ fitness} = 4 \\ P1 &: 1111000100 \text{ fitness} = 5 \\ P2 &: 0101010000 \text{ fitness} = 3 \\ P3 &: 1111001110 \text{ fitness} = 7 \end{aligned} \tag{C.6}$$

Mutation

In mutation, one or more genes of a chromosome change. Each chromosome is given a chance to undergo mutation by generating a random number between 0 and 1 for each of its genes. If the random number is less than the mutation probability, here 0.1, the value of the gene is flipped. A possible mutation is shown in equation (C.7). After mutation the new population is shown in equation (C.8).

$$\begin{aligned} &\text{population} = P0, \text{ gene} = 4, \text{ new population} = 0001110101, \text{ fitness} = 5 \\ &\text{population} = P0, \text{ gene} = 6, \text{ new population} = 0001100101, \text{ fitness} = 5 \\ &\text{population} = P1, \text{ gene} = 5, \text{ new population} = 1111100100, \text{ fitness} = 6 \\ &\text{population} = P1, \text{ gene} = 9, \text{ new population} = 1111100110, \text{ fitness} = 7 \\ &\text{population} = P2, \text{ gene} = 9, \text{ new population} = 0101010010, \text{ fitness} = 4 \\ &\text{population} = P2, \text{ gene} = 6, \text{ new population} = 0101000010, \text{ fitness} = 3 \\ &\text{population} = P3, \text{ gene} = 6, \text{ new population} = 1111011110, \text{ fitness} = 8 \end{aligned} \tag{C.7}$$

$$\begin{aligned} P0 &: 0001100101 \text{ fitness} = 4 \\ P1 &: 1111100110 \text{ fitness} = 7 \\ P2 &: 0101000010 \text{ fitness} = 3 \\ P3 &: 1111011110 \text{ fitness} = 8 \end{aligned} \tag{C.8}$$

This completes the first generation. We observe that the average fitness of the population has increased. The evolution continues for the maximum number of generations or until the termination condition is met. In the final population the most fit population (the chromosome with the highest fitness) is the solution of the problem. Obviously for a given problem there may be more than one optimum solution.

Appendix D

Graphs of Queuing Simulations

Section 6.3 represented queuing performances of several traces for different values of H and R for $\rho = 0.7$. This appendix shows queuing performances of other traces for different values of ρ. These graphs are consistent with the results obtained in section 6.3.3. These results indicate that traces with similar R values have similar queuing performances at different utilization level.

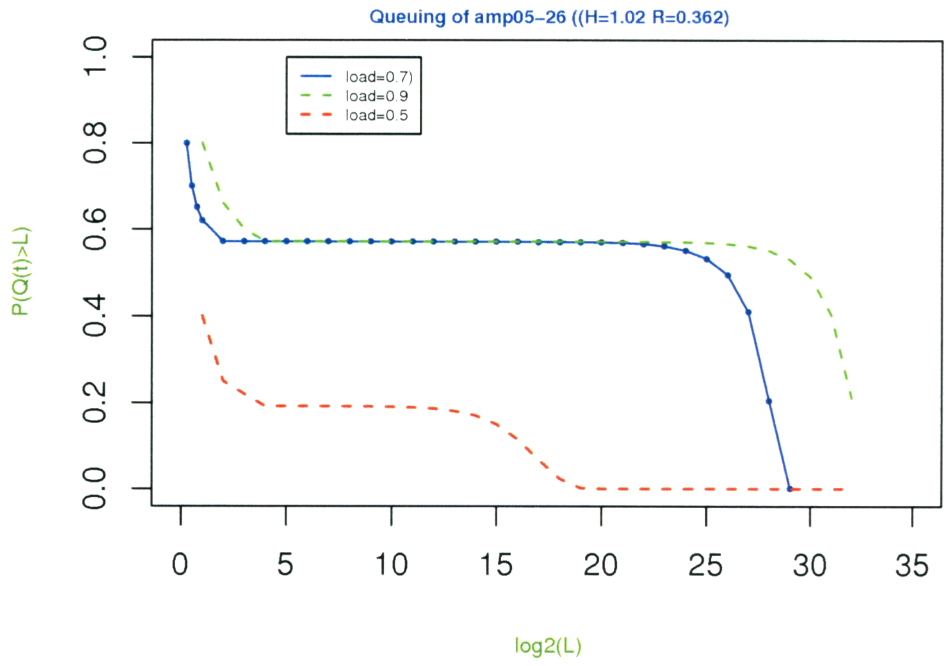


Figure D.1: Queuing Performance of AMP05-26 Trace at Different Loads.

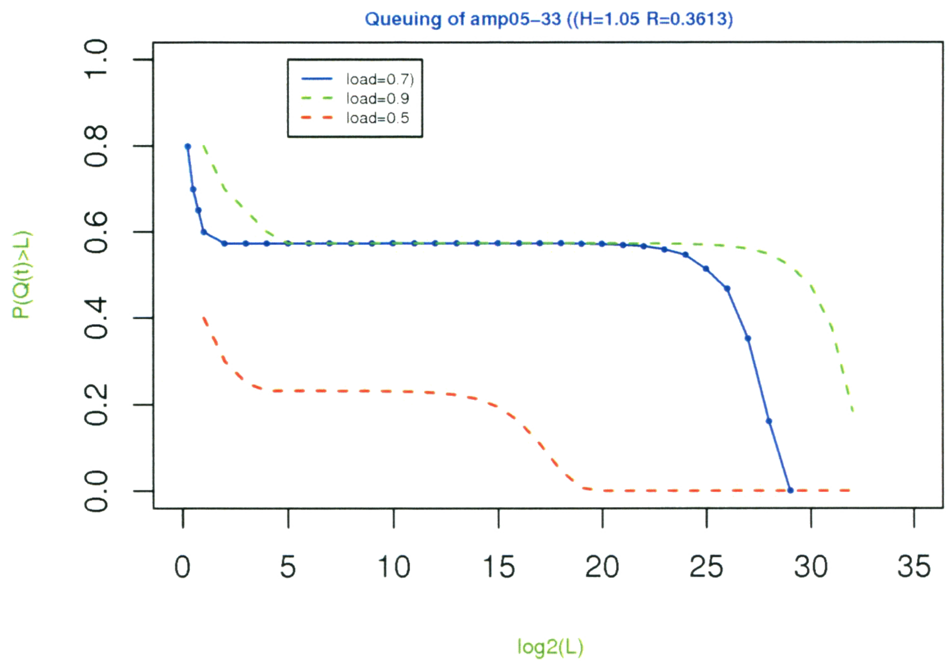


Figure D.2: Queuing Performance of AMP05-33 Trace at Different Loads.

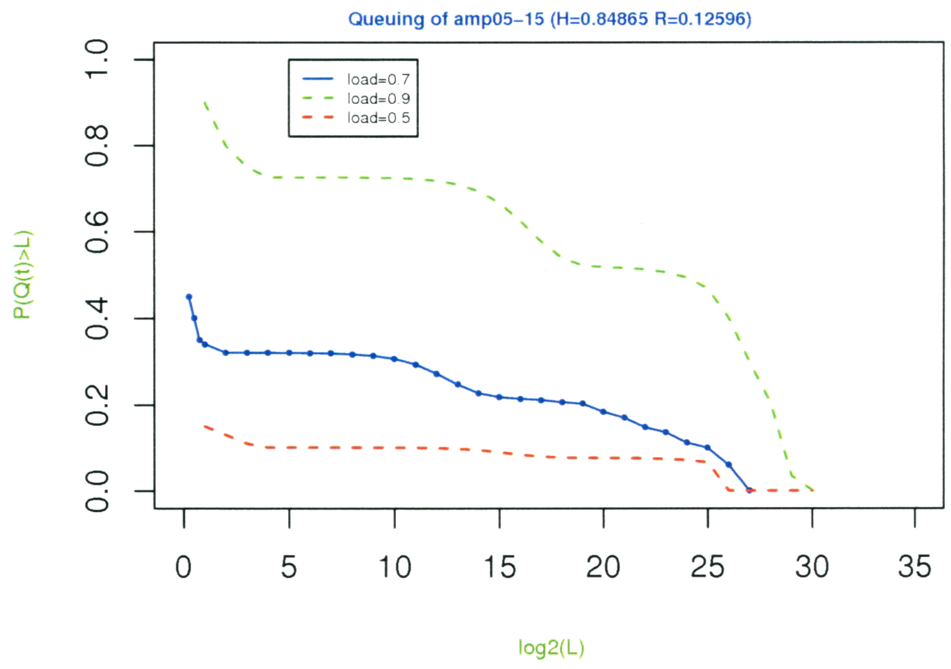


Figure D.3: Queuing Performance of AMP05-15 Trace at Different Loads.

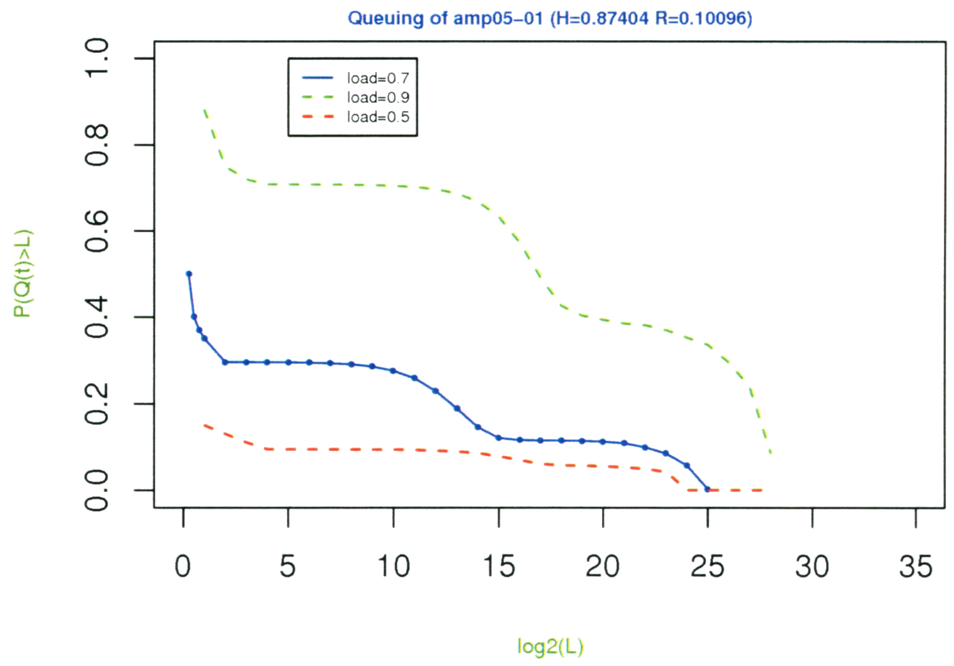


Figure D.4: Queuing Performance of AMP05-01 Trace at Different Loads.

Appendix E

Pseudocodes and Scripts

E.1 Pseudocode to Compute R

```
GET traffic_data, duration, link_speed
CALCULATE total_data as sum of sizes of all packets
DETERMINE total_packets as total number of packets
CALCULATE utilization=(total_data)/(duration×link_speed)
CALCULATE threshold_tgroup=(duration×utilization)/total_packets
DEFINE SMALL, MEDIUM and LARGE fuzzy sets for groupData
DEFINE SMALL, MEDIUM and LARGE fuzzy sets for groupTime
DEFINE SMALL, MEDIUM and LARGE fuzzy sets for R
SET numerator:=0
SET denominator:=0
FOR each packet of the trace
   READ packet_arrival_time
   IF (current_packet_arrival_time-previous_packet_arrival_time<
                                                threshold_tgroup)
        IF inGroup is FALSE
           SET inGroup to TRUE
           groupStartTime:=previous_packet_arrival_time
           groupData:=previous_packet_data
        ENDIF
        ADD current_packet_data to groupData
   ELSE
        IF inGroup is TRUE
           groupTime:=groupStartTime-current_packet_arrival_time
           COMPUTE membership of groupData to SMALL, MEDIUM and LARGE
           COMPUTE membership of groupTime to SMALL, MEDIUM and LARGE
```

```
            COMPUTE membership of R to SMALL, MEDIUM and LARGE
            COMPUTE memsmall_R as membership of R to SMALL
            COMPUTE memmedium_R as membership of R to MEDIUM
            COMPUTE memlarge_R as membership of R to LARGE
            ADD memsmall_R×sup_small_R to numerator
            ADD memmedium_R×sup_medium_R to numerator
            ADD memlarge_R×sup_large_R to numerator
            ADD memsmall_R, memmedium_R and memlarge_R to denominator
            SET inGroup to FALSE
            groupData:=0
        ENDIF
    ENDIF
ENDFOR
COMPUTE R:=$\frac{numerator}{denominator}$
```

E.2 Computing R Using Perl

E.2.1 The Script

```
#!/usr/bin/perl
use Mysql;
use List::Util qw[min max];
$tbname=$ARGV[0];
$duration=$ARGV[1];
$host="localhost";
$db="traffic";
$user="user";
$pass="pass";
$db=Mysql->connect($host,$db, $user, $pass);
$rsql="select util from tracestat where trace_url
like '%$tbname%' limit 1";
$rqry=$db->query($rsql) or print "$rsql\n";
while (@rrow=$rqry->fetchrow) {
   $util=$rrow[0];
}
$betasql="select (sum(length)*48)/count(*)
from $tbname";
```

```
$betaqry=$db->query($betasql);
while (@betarow=$betaqry->fetchrow) {
       $beta=$betarow[0];
}
print "BETA: $beta bytes\n";
### TauG; ### TauG=((duration)/(total packet no))*util
$tausql="select (($duration*1000000)/count(*))
from $tbname";
$tauqry=$db->query($tausql)
    or print "$tausql\n";
while (@taurow=$tauqry->fetchrow) {
    $taug=$taurow[0];
}
print "TauG= $taug micro seconds\n";
### find s (service rate)
$ssql="select ((sum(length))/($duration*1000000))
/($util/100) from $tbname";
$sqry=$db->query($ssql) or print "$ssql\n";
while (@srow=$sqry->fetchrow) {
    $s=$srow[0];
}
print "s= $s micro seconds\n";
## find supports for SMALL, MEDIUM and LARGE for B
$bss=$beta/6;
$bsm=$beta/3;
$bsl=$beta/2;
## find supports for SMALL, MEDIUM and LARGE for delta
$dss=$bss/$s;
$dsm=$bsm/$s;
$dsl=$bsl/$s;
$rsql="select time_sec, time_micro_sec from
   $tbname order by autoid limit 1";
$rqry=$db->query($rsql) or print "$rsql\n";
while (@rrow=$rqry->fetchrow) {
  $isec=$rrow[0];
  $ptime=$rrow[1];
}
```

```
$tn=0;
$rsql="select time_sec, time_micro_sec,
  length, autoid from $tbname";
$rqry=$db->query($rsql) or print "$rsql\n";
while (@rrow=$rqry->fetchrow) {
$sec=$rrow[0];
$sec=$sec-$isec;
$msec=$rrow[1];
$time=$sec*1000000+$msec;
$autoid=$rrow[3];
## check time diff
if (($time-$ptime)<$taug) {
     if ($i<1) {
         $j=1;
         $groupstartid=$autoid-1;
         $startgroup=$ptime;
      }
  $i=1;
  if ($j>0) {
    #print "in group $rrow[2]...\n";
 }
  if ($j>1) { $groupbytes+=$rrow[2]; }
  $j++;
} ## end ($time-$ptime)<$taug
else {
  if ($i>0 && $j>5 ) {
     $grouptime=$ptime-$startgroup;
     $tn++;
     if ($groupbytes<=$bss) { $bsmall=1; }
     elsif ($groupbytes>=$bsm) { $bsmall=0; }
        $bsmall=($bsm-$groupbytes)/($bsm-$bss);
     }
     if ($groupbytes<=$bsm) { $blarge=0; }
     elsif ($groupbytes>=$bsl) { $blarge=1; }
     else {
       $blarge=($groupbytes-$bsm)/($bsl-$bsm);
     }
     if ($groupbytes<=$bss) { $bmedium=0; }
     elsif ($groupbytes>=$bsl) { $bmedium=0; }
     elsif (($groupbytes>$bss) && ($groupbytes<$bsm)){
          $bmedium=($groupbytes-$bss)/($bsm-$bss);
     }
     else {
        $bmedium=($bsl-$groupbytes)/($bsl-$bsm);
     }
     if ($grouptime<=$dss) { $dsmall=1; }
     elsif ($grouptime>=$dsm) { $dsmall=0; }
     else {
        $dsmall=($dsm-$grouptime)/($dsm-$dss);
     }
     if ($grouptime<=$dsm) { $dlarge=0; }
     elsif ($grouptime>=$dsl) { $dlarge=1; }
     else {
       $dlarge=($grouptime-$dsm)/($dsl-$dsm);
     }
if ($grouptime<=$dss) { $dmedium=0; }
elsif ($grouptime>=$dsl) { $dmedium=0; }
elsif (($grouptime>$dss) && ($grouptime<$dsm)){
  $dmedium=($grouptime-$dss)/($dsm-$dss);
} else {
       $dmedium=($dsl-$grouptime)/($dsl-$dsm);
}
if (($bsmall>0)&&($dsmall>0){
   $y= min($bsmall, $dsmall); push(@rsmall, $y);
}
if (($bsmall>0)&&($dmedium>0)){
  push(@rsmall, min($bsmall, $dmedium));
}
if (($bsmall>0)&&($dlarge>0)){
  push(@rsmall, min($bsmall, $dlarge));
}
if (($bmedium>0)&&($dsmall>0)){
  push(@rmedium, min($bmedium, $dsmall));
}
if (($bmedium>0)&&($dmedium>0)){
  push(@rsmall, min($bmedium, $dmedium));
}
if (($blarge>0)&&($dlarge>0)){
  push(@rsmall, min($bmedium, $dlarge));
}
if (($blarge>0)&&($dsmall>0)){
  push(@rhigh, min($blarge, $dsmall));
}
if (($blarge>0)&&($dmedium>0)){
  push(@rmedium, min($blarge, $dmedium));
}
if (($blarge>0)&&($dlarge>0)){
  push(@rsmall, min($blarge, $dlarge));
}
$groupbytes=0;
}  ## end ($i>0 && $j>3 )
$i=0;
}  ### end else
$byte=$rrow[2];
$diff=$time-$ptime;
$ptime=$time;
}
$sd=0; foreach (@rsmall) {
  $x=$_; $sn=$sn+$x*0.05; $sd=$sd+$x;
}
$md=0; $mn=0;
foreach (@rmedium) {
   $x=$_;
   $mn=$mn+$x*0.5;
   $md=$md+$x;
}
$hd=0; $hn=0;
foreach (@rhigh) {
 $x=$_; #$hn=$hn+$x*0.75;
 $hn=$hn+$x*1; $hd=$hd+$x;
}
```

```
$r=($sn+$mn+$hn)/($sd+$md+$hd);
print "R: $r\n"; print "tn: $tn\n";
$usql="update tracestat set rparam='$r' where trace_url like '%$tbname%'
$uqry=$db->query($usql);
```

E.2.2 Runtime

Table E.1: Time Required to Computer R for off-line traffic tarces

Trace Name	Total Packets	Total Bytes	Total T-groups	Time Required
amp05-39	3541707	2.61 GB	134660	43 Sec
amp05-38	3707165	2.52 GB	146868	42 Sec
amp05-32	6398300	3.96 GB	249612	45 Sec
amp05-27	8668304	4.64 GB	252450	110 Sec
amp05-01	769100	0.52 GB	3655	7 Sec

E.3 MATLAB script to compute the Hurst Parameter

```
Hurst_wavelet
% This script computes the Hurst parameter of a traffic trace
% using Abry and Veitch Method. This script plots the Spectral
% Density of a given series, which then used to compute the
% Hurst parameter of the series. First, the traffic data is
% loaded from a mysql database table to a text file as a
% sequence (using a perl script). The text file is used as the
% input file of this script. The sequence in the source is then
% loaded into the 1-D Wavelet Analyzer, and  analyzed using the
% a Daubechies wavelet. The wavelet coefficients are computed
% and stored.
% After generating coefficients, I find the variance at each
%scale and plot these variances.  Then use Least Squares
%Method to plot the slope of changing variance.  This is the
%method described by Abry and Veitch in "Wavelet Analysis of
%Long-Range Dependent  Traffic" IEEE Transactions on
%Information Theory, vol. 44, No. 1, Jan 1998.The Hurst
%parameter may next be calculated as H = 1/2*(slope+1).
% The wavelet coefficients are arranged in a concatenated
```

```
%vector,indexed by a 'longs' vector. The first # in longs
%corresponds with the indices of the averaged signal after
%7 levels of  decomposition. The next number corresponds
%to the indices of the level 7 wavelet coefficients, etc.
%The last number corresponds to the length of the
% original signal, and is not present in the coeffs
%vector. This breakdown is well explained at the bottom
%of "Importing and Exporting Information from the
%Graphical Interface", from the Help Menu of Wavelet Toolbox.
% Script originally written by LT Sam Edwards on
%21 July 2005 and is modified by A. Zia Rahman on
%January 4, 2006.
clear;
load byteout_detrend_coefs;
indices=longs;
a7=1:indices(1);
% Separate indices for each scale's coeffs.
b7=indices(1)+1:sum(indices(1:2));
b6=sum(indices(1:2))+1:sum(indices(1:3));
b5=sum(indices(1:3))+1:sum(indices(1:4));
b4=sum(indices(1:4))+1:sum(indices(1:5));
b3=sum(indices(1:5))+1:sum(indices(1:6));
b2=sum(indices(1:6))+1:sum(indices(1:7));
b1=sum(indices(1:7))+1:sum(indices(1:8));
scalevar=zeros(1,7);
% Calculate the variance at each scale
scalevar(1)=var(coefs(b1));
scalevar(2)=var(coefs(b2));
scalevar(3)=var(coefs(b3));
scalevar(4)=var(coefs(b4));
scalevar(5)=var(coefs(b5));
scalevar(6)=var(coefs(b6));
scalevar(7)=var(coefs(b7));
%stand=zeros(1,10);
% Now, convert to Standard Deviations at each scale
%stand(1)=sqrt(scalevar(1));
%stand(2)=sqrt(scalevar(2));
```

```
%stand(3)=sqrt(scalevar(3));
%stand(4)=sqrt(scalevar(4));
%stand(5)=sqrt(scalevar(5));
%stand(6)=sqrt(scalevar(6));
%stand(7)=sqrt(scalevar(7));
%stand(8)=sqrt(scalevar(8));
%stand(9)=sqrt(scalevar(9));
%stand(10)=sqrt(scalevar(10));
logvar=log2(scalevar);
logvar=logvar';
scales=1:7;
scales=scales';
[z1,slope] = LeastSquares(scales(1:7),logvar(1:7));
plot(scales,logvar,'*'); hold on;
plot(scales(1:7),z1,'r-'); hold off;
title(['Spectral Densities of Traffic byteout using db5 basis function,
slope is ', num2str(slope(1))])  xlabel('Scales')
ylabel('Log2(Spectral Densities)')
legend('Spectral Densities','Reconstructed Line')
```

E.4 Trace-Driven Queue Simulator

```
// A trace-driven queuing simulator using CSIM18 trace/trace/1/K
// queue simulation.The input file is a <delta_time, pkt_len> pairs.
// pkt_len is in bytes and delta_time is in seconds.SERVICE_RATE
// is the router service rate in bytes per second. CAPACITY is
// the router capacity in bytes. The script uses csim.h header file.
// The filehandle QF holds the queue size and time-stamp of
// instantaneous queue length Q(t).
void queue(int pkt_len)
{
     create("queue");
     // Increment total arriving counter
     Total_count++;
     // If no space for packet, discard. (log to overflow file)
```

```
        if (pkt_len > (CAPACITY - Bytes_buffered)) {
            Discard_count++;
            fprintf(Of, "%f -- %9d -- %5d \n",
                (clock - Last_over_time), Total_count, pkt_len);
            Last_over_time = clock;
            return;
        }
        // The packet got buffered, so update Bytes_buffered
        // and increment Accept
        Bytes_buffered = Bytes_buffered + pkt_len;
        Accept_count++;
        // Reserve, hold, and release server
        reserve(Server);
        hold((pkt_len) / SERVICE_RATE);
        release(Server);
        // Update Bytes_buffered
        Bytes_buffered = Bytes_buffered - pkt_len;
        fprintf(QF, "%f -- %9d\n",
                clock, Bytes_buffered);
        return;
    }
```

E.5 Perl Script to read 2005 AMP trace

```
#!/usr/local/bin/perl;
#
# The script takes the filename as command line input and
# inserts each packet parameters to a mysql database.
#
use Mysql;
$host="localhost";
$db="traffic";
$user="traffic";
$pass="traffic";
$db=Mysql->connect($host,$db, $user, $pass);
$file=$ARGV[0];
```

```
$file=~s/\.tsh//;
$tsql="drop table if exists $file";
$tqry=$db->query($tsql);
$tsql="create table $file (autoid double unsigned auto_increment
PRIMARY KEY, time_sec varchar(32), time_micro_sec varchar(32),
length mediumint unsigned, protocol smallint, source_address
varchar(50), dest_address varchar(50), source_port mediumint
unsigned, dest_port mediumint unsigned, seq_number int unsigned,
ack_number int unsigned, window_size mediumint unsigned)";
$tqry=$db->query($tsql);
## open the input AMP traffic in tsh format.
open(infile,$ARGV[0]) || die("Can't open input file\n");
stat($ARGV[0]);
$s=-s _;
$i=0;
read(infile,$record, $s);
$k=$s/44;
for($i=0; $i<$k; $i++)
{
    $j=$i*11;
    ## time-stamp of packet
    $sec=vec($record, $j,32);
    ## time-stamp of packet
    $ms=vec($record, $j+1,32);
    $intf=($ms>>24)&0xff;
    $ms=$ms&0xffffff;
    $plength=vec($record, $j+2,32);
    #$intf=($ms>>24)&0xff;
    $plength=$plength&0xffff;  ## packet length
    $protocol=vec($record, $j+4,32);  ## protocol
    $protocol=($protocol>>16)&0xffff;
    $protocol=$protocol&0xff;
    $sa=vec($record, $j+5,32);  ## source address
    $first=($sa>>24)&0xff;
    $second=($sa>>16)&0xff;
    $third=($sa>>8)&0xff;
    $fourth=$sa&0xff;
    ## source IP
    $sip="$first"."\."."$second"."\."."$third"."\."."$fourth";
    $da=vec($record, $j+6,32);   ## destination address
    $first=($da> >24)&0xff;
    $second=($da> >16)&0xff; $third=($da> >8)&0xff; $fourth=$da&0xff;
    ## destination IP
    $dip="$first"."\."."$second"."\."."$third"."\."."$fourth";
    $spdp=vec($record, $j+7,32);
    $sp=($spdp> >16)&0xffff; ## source port
    $dp=$spdp&0xffff;   ## destination port
    $seqno=vec($record, $j+8,32);    ## TCP sequence number
    $ackno=vec($record, $j+9,32);    ## TCP ack number
    $window=vec($record, $j+10,32);
    $window=$window&0xffff;          ## TCP window size
    $tsql="insert into $file values ('', '$sec', '$ms', '$plength',
     '$protocol', '$sip', '$dip', '$sp','$dp', '$seqno',
     '$ackno', '$window')";
    $tqry=$db->query($tsql);
}
```

E.6 Pseudocode for the Simulation of FAFRED

```
INITIALIZATION
    $r \leftarrow 0$
    $Q = 0$
FOR EACH packet arrival
    IF $Q < \frac{\beta}{m}$
        ENQUE packet
        COMPUTE $R$
        Update $RTable$
    ELSIF $Q = \beta$
        DROP packet
    ELSE
        COMPUTE $STATE$
        CASE $STATE$ OF
            QUIET          : $r = 0$
```

```
                BURST            : increase $r$ by $\eta$
                UPWARD ALARM     : increase $r$ by $\epsilon$
                ALARM            : increase $r$ by $\epsilon$
                GREEDY           : increase $r$ by $\eta$
                DOWNWARD ALARM   : decrease $r$ by $\epsilon$
                DOWNWARD QUIET   : $r = 0$
                CONSTANT         : $r$ is unchanged
            ENDCASE
            IF $r = 0$
                ENQUE packet
                COMPUTE $R$
                Update $RTable$

            ELSE
                COMPUTE $r_p \leftarrow \frac{r \times R}{avgr}$
                with probability $r_p$
                    DROP the arriving packet
                    SET $R \leftarrow 0$
            ENDIF
        ENDIF
    ENDFOR
    Fixed Parameters:
        $\epsilon$ : FAFRED parameter
        $\eta$ : FAFRED parameter
        $m$ : FAFRED parameter
        $\beta$ : buffer size
    Other:
        $Q$       : current queue length
        $R$       : $R$ parameter of the flow of arriving packet
        $avgr$    : average of the $R$ values of all the active flows
        $r$       : global drop probability
        $r_p$     : packet drop probability
        $STATE$   : linguistic variable that represents traffic state
```

Appendix F

List of Symbols

β	Buffer Capacity
ζ	Energy of a Poisson Process
λ	Arrival Rate
μ	Membership Function
ρ	Utilization
σ^2	Variance
ς	Gaussian White Noise
τ_G	T-Group Time-out Threshold
τ_f	f-Group Time-out Threshold
ψ	Energy of a Process
Γ	A traffic trace
C	Link Capacity
E	Expected Value
FI	Fairness Index
H	Hurst Parameter
G_i	T-Group
$Q(t)$	Queue Length at Time t
R	R Parameter
T	Round Trip Time
$X(t)$	Stochastic Process
W	Wavelet Coefficient
$Z(t)$	Fractional Brownian Process
$r(k)$	Auto-correlation Function of Lag k

g_i	f-Group
l	Packet Drop Rate
p	Packet Drop Probability
q	Average Queue Length
v	Average Queuing Time Variance
w_i	*ith* Weight Parameter

Appendix G

Traces Used

2005 AMP Traces

No	URL	Timestamp of Collection	TCP	UDP	Rest	Total Bytes	Total Packets	Total Connection	Util	H	R
amp05-01	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050311/AMP-1110523221-1.tsh.gz	1110523310	87.11	12.22	0.67	523992584	769100	39580	26.95	0.87404	0.10096
amp05-02	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050311/AMP-1110545630-1.tsh.gz	1110545720	84.37	14.605	1.025	2900093323	4233868	163236	24.864	0.82008	0.10942
amp05-03	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050311/AMP-1110564441-1.tsh.gz	1110564529	86.494	13.146	0.36	3443192926	5164425	173921	29.52	0.84816	0.12448
amp05-04	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050311/AMP-1110577446-1.tsh.gz	1110577537	82.09	17.71	0.2	2259430219	3562988	199675	19.371	0.78728	0.08614
amp05-05	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050312/AMP-1110586282-1.tsh.gz	1110586372	83.887	15.875	0.238	2551632858	3849280	192619	21.876	0.83066	0.09916
amp05-06	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050312/AMP-1110608691-1.tsh.gz	1110608782	82.069	17.762	0.169	2260299700	3365418	151318	19.378	0.82632	0.08451
amp05-07	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050312/AMP-1110631100-1.tsh.gz	1110631185	77.623	22.185	0.192	2101430825	3237514	154610	18.016	0.79223	0.08992
amp05-08	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050312/AMP-1110653509-1.tsh.gz	1110653600	77.964	21.771	0.266	1851495597	2748334	68710	15.874	0.80991	0.0882
amp05-09	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050312/AMP-1110662915-1.tsh.gz	1110663000	70.783	28.974	0.244	1391937984	2227360	86244	11.934	0.73258	0.0673
amp05-10	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050313/AMP-1110675351-1.tsh.gz	1110675442	81.623	18.199	0.178	2212237085	3384193	158100	18.966	0.82472	0.09475
amp05-11	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050313/AMP-1110707167-1.tsh.gz	1110707258	70.671	29.165	0.164	1303605930	10002842	86604	11.176	0.72236	0.08079
amp05-12	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050313/AMP-1110716571-1.tsh.gz	1110716662	57.894	41.609	0.497	1091931459	9454903	87867	9.362	0.69053	0.07686
amp05-13	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050313/AMP-1110729576-1.tsh.gz	1110729662	70.353	29.491	0.156	1256851439	10139421	153307	10.775	0.71241	0.08811
amp05-14	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050313/AMP-1110738981-1.tsh.gz	1110739072	75.022	23.819	1.159	1667377741	8598129	49708	14.295	0.80634	0.09703
amp05-15	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050313/AMP-1110748385-1.tsh.gz	1110748476	74.484	25.125	0.391	1993003128	6132624	68502	17.087	0.84865	0.12596
amp05-16	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050314/AMP-1110760823-1.tsh.gz	1110760909	80.268	19.588	0.143	1668658522	4174476	158635	14.306	0.85936	0.10288
amp05-17	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050314/AMP-1110770227-1.tsh.gz	1110770318	76.949	22.795	0.257	2104833361	5313018	111960	18.046	0.81893	0.11512
amp05-18	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050314/AMP-1110783232-1.tsh.gz	1110783323	86.168	13.069	0.763	3029393345	7992868	124808	25.972	0.77846	0.1963
amp05-19	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050314/AMP-1110792637-1.tsh.gz	1110792725	95.382	3.545	1.073	2991029863	8881071	151132	25.643	0.79794	0.25431
amp05-20	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050314/AMP-1110802041-1.tsh.gz	1110802131	84.303	5.824	9.872	2944264079	9609340	168655	25.242	0.84194	0.25133
amp05-21	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050314/AMP-1110815046-1.tsh.gz	1110815133	93.02	5.242	1.738	2787262063	8165544	155367	23.896	0.77516	0.22053
amp05-22	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050314/AMP-1110824452-1.tsh.gz	1110824542	68.734	2.83	28.435	3284660008	10439397	129744	28.161	0.80621	0.30928
amp05-23	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050315/AMP-1110855697-1.tsh.gz	1110855787	86.468	8.352	5.18	2914744580	6725337	66122	24.989	0.8538	0.13338
amp05-24	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050315/AMP-1110868702-1.tsh.gz	1110868793	95.449	3.192	1.358	3228204398	8393379	104827	27.677	0.76692	0.19352
amp05-25	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050315/AMP-1110878107-1.tsh.gz	1110878191	84.321	15.528	0.151	4307070911	7240151	272167	36.926	0.78712	0.21176
amp05-26	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050315/AMP-1110891112-1.tsh.gz	1110891202	61.804	5.119	33.077	4787087734	10881325	124115	41.042	1.02001	0.36235
amp05-27	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050315/AMP-1110900517-1.tsh.gz	1110900605	80.245	2.392	17.363	4638400527	8668304	123145	39.767	0.77436	0.27649
amp05-28	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050315/AMP-1110909919-1.tsh.gz	1110910009	57.815	7.739	34.445	4850718509	6834647	101437	41.587	0.68033	0.17491
amp05-29	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050316/AMP-1110931762-1.tsh.gz	1110931849	64.275	10.313	25.411	4046320730	5280534	75709	34.691	0.68338	0.13068
amp05-30	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050316/AMP-1110944767-1.tsh.gz	1110944857	77.72	22.053	0.227	1883610729	2793144	62857	16.149	0.82602	0.06707
amp05-31	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050316/AMP-1110954170-1.tsh.gz	1110954261	87.479	12.413	0.107	3549393391	4975222	122790	30.43	0.82822	0.11498
amp05-32	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050316/AMP-1110967176-1.tsh.gz	1110967259	85.107	14.617	0.275	3962675522	6398300	192637	33.974	0.85638	0.15624
amp05-33	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050316/AMP-1110976581-1.tsh.gz	1110976671	67.639	7.257	25.104	4583061503	10752787	135026	39.292	1.05011	0.3613
amp05-34	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050316/AMP-1110985986-1.tsh.gz	1110986077	62.734	2.164	35.102	3689570883	4730322	95291	31.632	0.68354	0.13367
amp05-35	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050317/AMP-1111052646-1.tsh.gz	1111052735	55.356	44.494	0.15	5276665038	8163524	138514	45.239	0.68727	0.27868
amp05-36	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050317/AMP-1111062051-1.tsh.gz	1111062140	56.042	21.102	22.856	4829964040	9332058	293331	41.409	0.69795	0.29962
amp05-37	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050317/AMP-1111075055-1.tsh.gz	1111075146	97.414	2.251	0.335	4655009627	6510191	133821	39.909	0.77797	0.17375
amp05-38	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050317/AMP-1111084460-1.tsh.gz	1111084545	96.236	3.618	0.146	2527818492	3707165	101407	21.672	0.88904	0.1029
amp05-39	http://pma.nlanr.net/Traces/Traces/long/apth/1/20050317/AMP-1111093864-1.tsh.gz	1111093954	98.238	1.596	0.166	2606063926	3541707	90719	22.343	0.87579	0.10166

Table 1: 2005 AMP Traces. A 10 days collection of randomly sampled 10 minute IP header traces collected at AMPATH, Miami, FL, in March 2005.

2003 Auckland Traces

No	Trace URL	TCP	UDP	Rest	Util	Packets/sec	Total Connections	H	R
Auck0301	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031208-000000.gz	76.33	23.30	0.36	7.88	1191	72015	0.89	0.048
Auck0302	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031208-060000.gz	76.23	23.51	0.26	2.62	598	28956	0.84	0.045
Auck0303	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031208-120000.gz	68.52	31.1	0.38	19.01	4199	161976	0.84	0.072
Auck0304	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031208-180000.gz	87.79	12	0.21	16.92	2812	11993	0.85	0.074
Auck0305	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031202-000000.gz	83.59	16.06	0.35	6.38	923	68734	0.86	0.046
Auck0306	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031202-060000.gz	78.26	21.43	0.31	2.12	399	25879	0.76	0.043
Auck0307	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031202-120000.gz	74.05	25.63	0.32	15.01	4008	142980	0.83	0.071
Auck0308	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031202-180000.gz	89.78	9.98	0.24	13.97	2451	109976	0.91	0.082
Auck0309	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031203-000000.gz	71.32	28.35	0.33	5.56	1008	66129	0.78	0.059
Auck0310	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031203-060000.gz	70.61	29.13	0.26	2.77	429	16723	0.84	0.036
Auck0311	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031203-120000.gz	83.88	15.74	0.38	18.01	3887	147987	0.87	0.062
Auck0312	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031203-180000.gz	68.09	31.79	0.12	14.03	2398	107871	0.79	0.073
Auck0313	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031204-000000.gz	79.07	20.81	0.12	6.77	911	77899	0.87	0.055
Auck0314	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031204-060000.gz	74.41	25.43	0.16	2.11	469	18225	0.82	0.040
Auck0315	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031204-120000.gz	79.67	20.22	0.11	15.98	3788	143993	0.81	0.074
Auck0316	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031204-180000.gz	85.05	14.84	0.11	14.99	2187	103912	0.92	0.079
Auck0317	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031205-000000.gz	83.55	16.18	0.27	6.01	991	59897	0.88	0.066
Auck0318	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031205-060000.gz	68.20	31.46	0.34	2.12	487	25125	0.87	0.056
Auck0319	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031205-120000.gz	88.08	11.66	0.26	19.87	12190	148653	0.77	0.075
Auck0320	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031205-180000.gz	83.54	16.21	0.25	15.12	13185	113343	0.81	0.063
Auck0321	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031206-000000.gz	75.42	24.32	0.26	2.99	821	46124	0.87	0.019
Auck0322	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031206-060000.gz	88.23	11.65	0.12	1.21	349	11545	0.87	0.012
Auck0323	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031206-120000.gz	68.38	31.51	0.11	11.02	1761	99767	0.77	0.042
Auck0324	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031206-180000.gz	78.08	21.75	0.17	10.34	1209	85120	0.84	0.057
Auck0325	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031207-000000.gz	84.14	15.57	0.29	3.33	602	51201	0.85	0.012
Auck0326	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031207-060000.gz	85.2	14.54	0.26	1.12	298	12498	0.82	0.015
Auck0327	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031207-120000.gz	82.64	17.22	0.14	11.66	1518	97012	0.89	0.065
Auck0328	http://pma.nlanr.net/Traces/Traces/long/auck/8/20031207-180000.gz	73.27	26.49	0.24	12.21	1674	87024	0.88	0.049

Table 2: 2003 Traces. The original traces are 24 hour long. The statistics are for truncated traces of 20 min long.

2001 Auckland Traces

No	Collection Time	TCP	UDP	Rest	Util	Pacets/Sec	Total Connections	H	R
Auck0101	Mon00	94.53	5.28	0.19	1.88	412	15093	0.98	0.013
Auck0102	Mon06	98.35	1.34	0.31	2.02	367	18956	0.92	0.014
Auck0103	Mon12	98.29	1.34	0.37	6.01	1423	36897	0.92	0.052
Auck0104	Mon18	91.13	8.77	0.10	5.42	1576	37993	0.86	0.033
Auck0105	Tue00	92.74	7.15	0.11	1.88	379	8734	0.96	0.021
Auck0106	Tue06	93.18	6.70	0.12	1.12	511	5879	0.89	0.026
Auck0107	Tue12	91.78	7.93	0.29	4.01	1789	32980	0.87	0.041
Auck0108	Tue18	95.21	4.64	0.15	5.98	1820	34976	0.89	0.047
Auck0109	Wed00	91.53	8.24	0.19	3.56	423	6129	0.84	0.021
Auck0110	Wed06	97.18	2.55	0.27	1.77	310	6723	0.84	0.016
Auck0111	Wed12	92.53	7.34	0.13	6.01	1412	27987	0.87	0.042
Auck0112	Wed18	97.17	2.51	0.32	3.03	1457	35871	0.93	0.033
Auck0113	Thu00	95.65	4.14	0.21	1.77	398	11899	0.88	0.011
Auck0114	Thu06	93.22	6.63	0.15	2.11	422	8222	0.89	0.021
Auck0115	Thu12	94.75	4.99	0.26	5.98	1456	33898	0.93	0.044
Auck0116	Thu18	94.47	5.32	0.21	4.99	1477	36765	0.95	0.054
Auck0117	Fri00	94.89	4.85	0.26	2.01	455	9897	0.96	0.026
Auck0118	Fri06	98.49	1.39	0.12	1.12	467	18125	0.93	0.036
Auck0119	Fri12	95.82	4.07	0.11	4.87	1398	38656	0.89	0.045
Auck0120	Fri18	90.10	9.76	0.14	5.12	1423	35783	0.88	0.053
Auck0121	Sat00	98.74	0.94	0.32	0.99	488	6124	0.96	0.011
Auck022	Sat06	95.79	4.09	0.12	1.01	564	6545	0.89	0.011
Auck0123	Sat12	94.51	5.13	0.36	4.02	1498	24767	0.85	0.018
Auck0124	Sat18	93.37	6.40	0.23	5.34	1378	25120	0.87	0.017
Auck0125	Sun00	94.40	5.49	0.11	0.83	321	7201	0.89	0.012
Auck0126	Sun06	93.72	6.07	0.21	0.72	345	6498	0.87	0.01
Auck0127	Sun12	96.53	3.37	0.10	4.66	1189	23452	0.94	0.031
Auck0128	Sun18	95.30	4.44	0.26	5.21	1209	24988	0.89	0.024

Table 3: 2001 Traces. The original traces are 24 hour long. The statistics are for truncated traces of 20 min long.