Self-Similar Traffic Modeling in Computer Communication Networks

Vikas Paliwal

December 7, 2003

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere thanks to Prof. Dimitrios Makrakis for providing me the invaluable guidance and regular feedback while I was working on this project. I also benefited immensely from the discussions during the classes of the course, *ELG5374(Computer Communication Networks)*. My thanks are also due to all the students in the course.

Lots of people have helped me to understand computer communication networks; in particular, I would like to mention my supervisor, Prof. Ioannis Lambadaris, Prof. Biswajit Nandy and Dr. Parsa R. Larijani, who constantly provided me the material and useful suggestions ever since I ventured into this field.

Chapter 1

Introduction

1.1 Computer Networks

Computer networks have been in the scene since the last 3-4 decades. With ever-growing internet and other networks, interest in computer networks is natural. More and more universities and research labs across the world are working towards enhancing the performance of computer networks. All this indicates towards a future with a greater role for networked computers. This has manifested in the form of various networked applications like internet surfing, distributed computing, etc. Industry, together with the research community, is welcoming this advent of computer networks with greater support as can be seen in *Sun Microsystems'* punch line, "Network is the computer".[1]

However, with growing popularity of networks, the volume of traffic on the network becomes an issue. As the number of users on a network (say, Internet) increases, traffic on the network is also bound to increase. A greater traffic on the network has its own implications on the network performance. Congestion, packet losses, greater delays in transmission are some of the phenomena that are bound to happen with increased network traffic.

On the flip side of it, the expectations of the network users are also increasing with time. These days applications require greater Quality of Service(QoS). To offer greater quality of service, network is expected to have greater throughputs, reduced packed losses and smaller delays.

All this makes the life of a modern-day network designer very difficult.

[1]It was, in fact, the main vision of Sun Microsystems' Chief Scientist, Bill Joy.

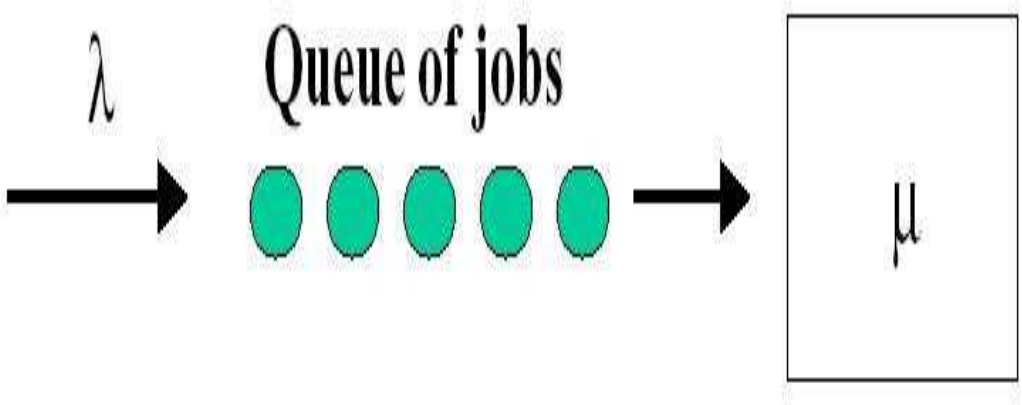


Figure 1.1: A Simple Queue

He has to meet two contrasting targets of support for greater traffic and enhanced QoS. This indicates towards serious planning on the part of a network designer. Models and simulators for networks are useful in predicting the network performance and hence are helpful in enhancing it. So, there is a growing need to develop accurate and fast models for modeling the computer networks[2, 10, 11].

1.2 A Simple Queue

A computer network can be studied at various levels. One standard way of looking at it is in terms of queuing theory, wherein, the offered network traffic load is studied at various nodes, viz. gateways, routers, servers and clients. A node can receive packets from various other nodes connected to it. Similarly, it can transmit packets to other nodes connected to it. The packet arrivals at a node usually indicate some trend and may be modeled using some statistical distribution. Upon arrival at a node a packet enters the queue and is being serviced by the node based on some queuing policy.

In a practical network, at each node some processing of the incoming packet is desirable. This amounts to a servicing time at each node and causes some delay. Fig. 1.1 shows a simple queue with, λ = arrival rate,

μ^{-1} = mean service time. Such a queue has traditionally been denoted with the notation, G/G/n, where the first two symbols stand for the arrival and service distributions, respectively. The last value denotes the number of servers. This is a simplistic model of a queue that can be used in network modeling at various nodes.

Analysis is available in the existing literature to calculate the various quantities of interest for such a system, viz. mean queue length, average waiting time, server utilization etc. However, such an analysis has so far been carried out in a simple framework with some distributional assumptions.

In the network domain such a queue can be thought of as packet queue at various nodes. Nonetheless, it is important to mention that modeling can only be done for the network after getting sufficient empirical data and then laying some distributional assumptions based on that. The most critical issues to be considered while modeling the network behavior are accuracy and speed. Whereas, accuracy for the model comes from logical assumptions for distributions based on empirical data and speed is inherent in the algorithm used for simulating the arrival and departure process.

1.3 The Failure of Poisson Modeling

Traditionally, for the sake of mathematical tractability and simplicity, Poisson arrivals are assumed in wide-area networks. A Poisson process with exponential inter-arrival times offers interesting properties. However, it has been shown that such a model is incapable of capturing the inter arrival behavior of wide-area and local-area networks [13, 14].

The interest in self-similar traffic modeling began after empirical results on traffic behavior in ethernet traffic were presented in [13]. This revolutionary paper based on data collected from traffic in the ethernet at Bellcore Morris Research Engineering Centre (MRE). This paper established through experimental results that interarrival behavior of packets differs from exponential. Prior to this point of time, network traffic was thought to be similar to telephone traffic where the Poisson model is of greater relevance and utility. However, this work presented strong arguments in favor of a different way of modeling the network traffic. There were greater research efforts in this direction to work out a suitable modeling approach for self-similar traffic.

This work showed that LAN traffic follows some *fractal*-like behavior where we have long-term *spikes*,*ripples* and *swells* [12]. Such a behavior calls

for a different treatment for this kind of behavior. In the coming section, we will try to understand this concept of self-similarity in greater details. Then we will proceed to look at some recent modeling techniques and a few measurement techniques. Also, a case study of one of the modeling technique that is used will be presented and a fuller analysis and implementation details of the approach will be detailed in later chapters.

Chapter 2

Understanding Self-Similarity

2.1 Pictorial View of Self-Similarity

A simple way of understanding self-similarity is in terms of scale-invariance. Basically it means that whatever be the time-scale for the traffic plots on a time scale, the plots will appear intuitively very "similar" to one another. An example of this concept is shown in Fig. 2.1, where starting from a time scale of 100s, all subsequent plots are obtained from the previous one by increasing the time resolution by a factor of 10. It is apparent that all the plots appear almost similar and exhibit self-similarity.

2.2 Definitions

A rigorous mathematical definition for self-similarity at this point is in place. Some widely used definitions for self-similarity are[19]:

1. Let $\{X_k\}$, $k = 0, 1, 2, \ldots$, be a discrete-time stationary process with mean μ, variance σ^2, and auto-correlation function $\{\rho_k\}$, for $k = 0, 1, 2, \ldots$, and let $\{X_k^{(m)}\}_{k=1}^{\infty} = \{X_1^{(m)}, X_1^{(m)}, \ldots, \}, m = 1, 2, 3, \ldots$, be a sequence of batch means, i. e. , $X_k^{(m)} = (X_{km-m+1} + \ldots + X_{km})/m, k \geq 1$.

   The process $\{X_k\}$ with $\rho^k \rightarrow k^{-\beta}$, as $k \rightarrow \infty, 0 \leq \beta \leq 1$, is called *exactly self-similar* with Hurst parameter, $H = 1 - (\beta/2)$, if $\rho_k^{(m)} = \rho_k$, for any $m = 1, 2, 3, \ldots$. In other words, the process, $\{X_k\}$ and the averaged process $\{X_k^{(m)}\}, m \geq 1$, have identical correlation structure.

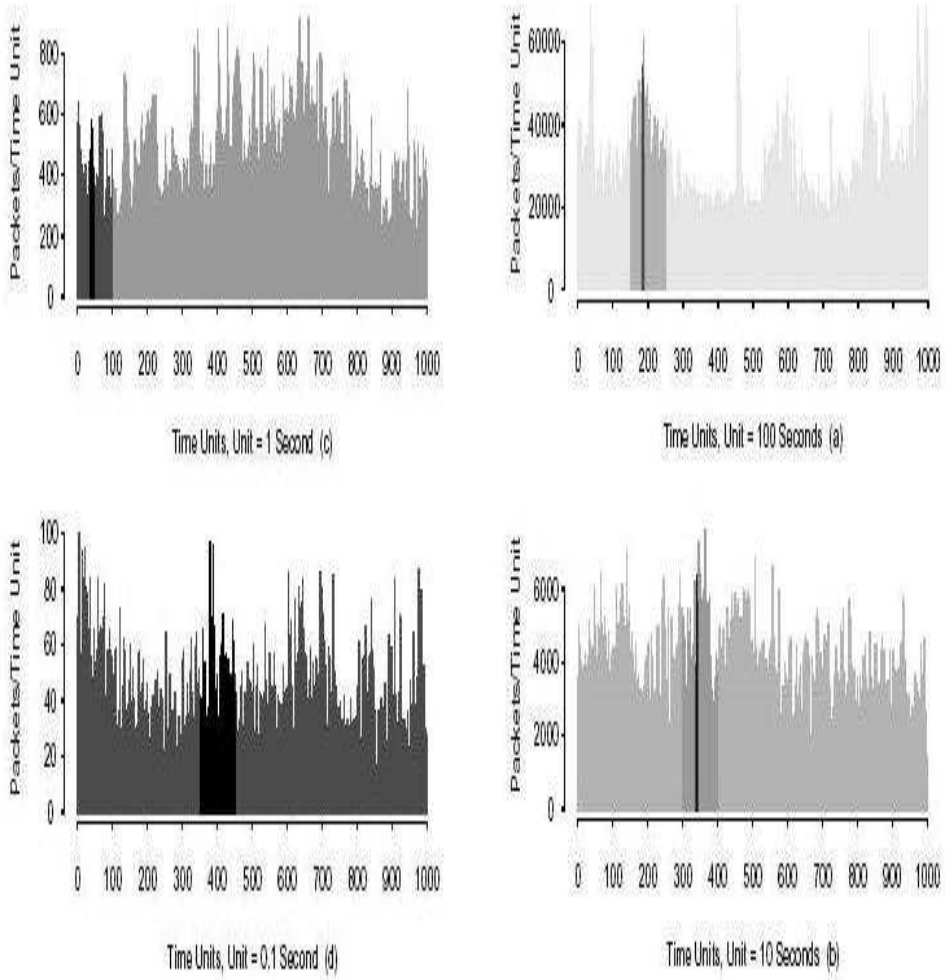


Figure 2.1: Self-Similarity in Picture (from [13])

The process $\{X_k\}$ is *asymptotically self-similar* with $H = 1 - (\beta/2)$, if $\rho_k^{(m)} \to \rho_k$, as $m \to \infty$.

2. A continuous time stochastic process $\{X_t\}$ is strongly self-similar with, $H(0 < H < 1)$, if for any positive stretching factor c, the rescaled process with time scale $ct, c^{-H}X_{ct}$ is equal in distributional sense to the original process, $\{X_t\}$. This implies that, for any sequence of time points $t_1, t_2, \ldots, t_n$, and for all $c > 0$, $\{c^{-H}X_{ct_1}, c^{-H}X_{ct_2}, \ldots, c^{-H}X_{ct_n}\}$ has the same distribution as $\{X_{t_1}, X_{t_2}, \ldots, X_{t_n}\}$.

3. In an engineering sense, the queue for a node exhibiting self-similarity will have heavy-tailed service-time distribution like,

$$P[X \geq x] \sim cx^{-\alpha}, x \to \infty, 0 \leq \alpha \leq 2 \tag{2.1}$$

If such a traffic load is offered to the queue with heavy-tailed service distribution, self-similarity in queue length process is observed. This fact has been used in the modeling of self-similar traffic in [15] and forms the basis of this work.

2.3 Main Properties

Main properties of self-similar processes include:

- *Slowly decaying variances* Usually for most other processes, the variance of the sample mean decreases along with the sample size. However, in case of self-similar process, the variance does not decrease as fast as sample size. Specifically, the variance experiences hyperbolic kind of decay instead of the faster exponential decay, i. e. , $Var[\{X_k^{(m)}\}] \to c_1 m^{-\beta_1}$ as $m \to \infty$, where c_1 is a positive constant and $0 < \beta_1 < 1$.

- *Long-range dependence* This property is manifested in a non-summable autocorrelation function. The summation of autocorrelations, $\Sigma_{k=0}^{\infty}\rho_k = \infty$. Again, the problem comes because of a hyperbolic decay instead of an exponential decay[8].

- *1/f-noise* The spectral density obeys a near-zero power law, i. e. , $f(\lambda, H) \to c_3\lambda^{1-2H}$, as $\lambda \to 0$. This is expected as long-range dependence in time-domain will have greater impact on the performance near zero point of the spectral response.

Mathematical impacts apart, self-similarity has greater impact on the performance of the network that we will see in the next chapter.

Chapter 3

Impact of Self-Similarity on Network Performance

3.1 Protocol View

In the sense of network protocols, the self-similarity comes from the length of sessions. Specifically, it has been shown that file transfers with files of sizes following a heavy-tailed distribution result in self-similarity. Self-similarity is more a phenomenon arising out of higher layer issues than lower layers. In fact, application layer parameters decide upon whether there would be self-similarity in the traffic behavior or not.

Application layer parameters like file sizes have greater impact on the transport layer [9]. Transport layer experiences the self-similarity effect when being offered a heavy-tailed load at the application layer. One way of analyzing the performance impact of self-similarity is to look in terms of sizes of files being transferred at the application layer as proposed in [23]. For the purpose of generation of a heavy-tailed distributed file sizes, pareto distributions have been used. At the transport layer, the effects are more pronounced in terms of congestion whereas at the lower layer it is seen in terms of self-similarity.

In a nutshell, all layers of the protocol architecture are affected with self-similarity in some or the other manner. This is expected as the root of it lies at the apex of protocol stack at the application layer which in turn gets transferred to the lower layers. 3.1 summarizes the impact on various layers:

Layer No.	Layer	Impact/Cause
5	Application	Heavy-tailed file-sizes
4	Transport	Congestion
3	Network	Greater Queueing Delays
2	Data Link Layer	Self-Similar Link Traffic
1	Physical	Greater bandwidth hogged

Table 3.1: Protocol Stack Architecture

3.2 Impacts

Self-similarity calls for greater network resources like link bandwidth and buffer space. Also, network performance goes down as seen by reduced throughput, greater packet loss rate and greater packet retransmissions. The overall network experiences reduced performance because of the long-range dependence.

3.2.1 Packet loss

A heavy-tailed service distribution as in self-similar traffic results in greater time spent by packet in queues at the nodes of the network. These nodes adhere to some queuing policy based on which they decide upon which packets to drop in case the buffer overflows. Whatever be the queuing mechanism viz. droptail, RED or ARED, sustained packet transfer result in greater waiting time for incipient packets in the queue. Upon overflow of the buffer, the router may even drop these packets. So it is clear that a long-term sustained flow results in greater loss for other flows. This is how self-similarity adversely affects the network and results in greater packet loss.

A greater problem coming out of this packet loss phenomena is packet retransmission in cases where the flows are connection-oriented like TCP. In such flows, loss of a packet results in retransmission of the entire packet. This has more pronounced effect as this results in greater traffic and an even greater packet losses. So the effect of self-similarity is compounded by the retransmission of packets.

3.2.2 Throughput

Throughput over a link is defined by the equation,

$$Throughput = \frac{N_T \times SS}{T}(bps) \tag{3.1}$$

where N_T is the number of packets transmitted in time T, SS is the segment size in bits.

Under the assumption of infinite link bandwidth and no loss, TCP throughput is,

$$Throughput_{max} = \frac{W \times SS}{RTT}(bps) \tag{3.2}$$

where W is TCP segment size and RTT is the round trip time. However, the exact behavior is deteriorated by packet losses leading to retransmissions. The effective throughput is considerably reduced owing to this and Eq. 3.2 represents only the raw throughput which is significantly reduced because of the packet losses. It is now clear that significant loss in throughput is experienced because of packet losses due to self-similarity. However analytical derivation of the throughput decline is involving and cumbersome and has been modeled in form of simulations only.

3.2.3 Queue Length

Because of sustained transfers, packets experience greater queuing delay than normal. This causes greater queue lengths and greater queuing delay. Fig. 3.1 shows the impact of increasing self-similarity (by reducing the pareto distribution's parameter, α) on the mean queue length in a simulation setup in [23].

3.3 Conclusion

As seen, self-similarity has pronounced impacts on the network performance. It hogs the bandwidth, causes packet delay and causes greater mean queue lengths at bottleneck links and hence is a major issue in network performance.

An effective design strategy for handling self-similar traffic would encompass greater buffer capacities and increased link capacities. However, doing so needs needs an effective modeling strategy so that based on simulations, design choices about network components can be made in a more effective fashion.

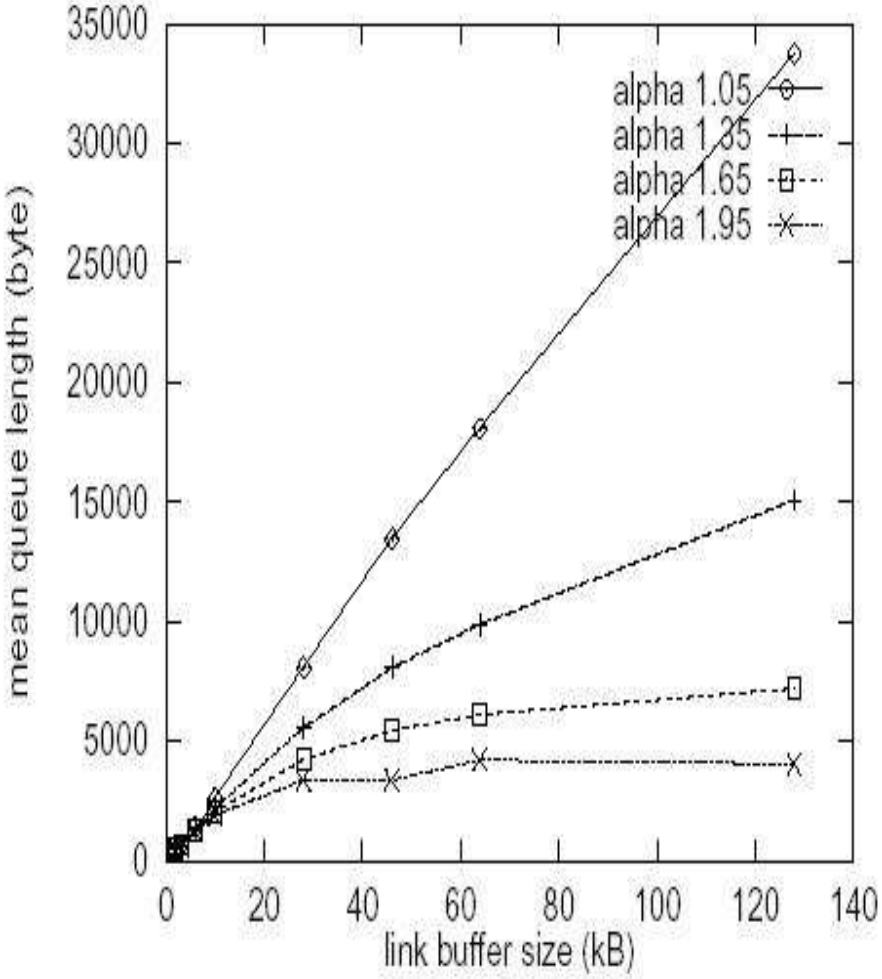


Figure 3.1: Effect of Self-Similarity on Mean Queue Length (from [23])

Chapter 4

Self-Similar Traffic Modeling

4.1 Introduction

As has been discussed in earlier chapters, modeling of self-similar traffic is critical to improving the performance of a network. Ever since it became clear that Poisson models are incapable of capturing the self-similar behavior in network traffic, researchers have been trying to come up with a suitable modeling strategy[17, 26, 27, 29]. Recently , many new attractive approaches have been proposed and have been shown to offer significant modeling flexibility.

Before delving into some of the most popular modeling techniques, it would be worthwhile for us to look into some expected features of a suitable modeling strategy.

- *Mean* A modeling strategy should be such that the empirical sample mean calculated from the traffic data, $\hat{\mu}$ matches fairly well with the mean of the modeling approach, μ.
- *Variance* For a matching till the second order, the sample variance of the empirical network traffic data, $\hat{\sigma^2}$ matches fairly well with the model variance, σ^2.
- *Self-Similarity* A suitable model should be able to appropriately capture the observed self-similarity in the empirical traffic data. This has to be verified by matching the estimate of the Hurst parameter, $\hat{H}$, from the traffic data and the Hurst parameter for the model, H.

- *Visual Similarity* The traffic data and the model when plotted as a time-series should appear similar to each other. There is clear measure for this feature but a modeler, based on his experience can check whether the model and the data follow the same path or not.

As clear from above, a model needs to meet some specific requirements. The utility of a model can be seen in terms of the parameters as mentioned above. There are effective statistical techniques to verify the correctness of a model that we will be seeing in the next chapter. Now we begin to look into some of the most recent and effective modeling strategies for modeling the self-similar network traffic behavior [2, 33, 34].

4.2 Fractional Gaussian Noise

Fractional gaussian noise generated using normal procedures is not capable of generating traffic with lower values of Hurst parameter, H. A popular variant of the approach presented in [22] is very fast and generates quality samples for a broad range of Hurst parameter. The algorithm for generating the traffic data using this approach is:

1. A sequence of values $\{f_1, f_2, \ldots, f_{n/2}\}$ is constructed, corresponding to power spectrum of an FGN process.

2. Each of the samples is fuzzed to get $\{\hat{f}_i\}$ so that the mean of actual power and the model are equal.

3. Then a sequence of complex numbers, $\{z_1, z_2, \ldots, z_{n/2}\}$ is generated so that it becomes a frequency-domain sample path. It is ensured that different sample paths generated using this method will be independent.

4. Expansion of the samples is done to produce spread them over a double of the earlier value, i. e. n. Now the signal corresponds to Fourier transform of the real-valued signal.

5. Finally, inverse fourier transform is performed to get the approximate time-domain FGN sample path.

The key factor of this approach is the use of the fast Fourier transform (FFT) for obtaining the sample. It is clear that FFT based algorithm are very

fast compared to others and offer a great computational advantage. It has been shown in the literature that generating the samples using this approach takes far less time than other popular approaches. This computational speed advantage makes it a very fast algorithm.

It is notable that for this approach, the mean, variance and the Hurst paramater H, are all independent and hence any kind of linear transformation can be preformed on the sample while preserving the value of H. In fact, it generated a sample path with zero mean sample path that needs to be transformed to a desired value. So, negative values may be eliminated from the model after suitable transformation with desired mean and variance. However, if even after doing so, the negative values do persist, then overall efficacy of *fGn* approach in modeling such a network traffic behavior becomes suspect. Under such circumstances, it is preferable to switch to another suitable modeling mechanism like fractional ARIMA (*fARIMA*) that is presented in the next section. So it is apparent that *fGn* processes correspond only to a specific class of self-similar process. There may be cases where it is unable to model the self-similar behavior of the network traffic. It has been shown that fractional Gaussian noise has a autocorrelation function as,

$$\tau(k) = 1/2(|k+1|^{2H} - |k|^{2H} + |k-1|^{2H}), k > 0 \tag{4.1}$$

Also it can be seen that *fGn* is exactly second-order self-similar with self-similarity parameter, H as long as $1/2 < H < 1$. Another popular variant of the fractional gaussian noise is the fractional Brownian motion (*fBm*). The only difference between *fGn* and *fBm* is that *fBm* is simply the sum of the fGn increments. In a sense, *fBm* is integrated version of *fGn*. In a nutshell, *fGn* processes are a wide class of process that are useful in developing a model for self-similar traffic in fast and scalable manner.

4.3 ARIMA Processes

ARIMA stands for Autoregressive Integrated Moving Average Processes and are named so as they are composed of following components:

- **AR(AutoRegressive)** This corresponds to the technique wherein the current sample is produced as linear combination of previous samples. For instance, an **AR** process of order p, refers to a process $\{x_n\}$ of

type:

$$x_n = e_n + a_1 x_{n-1} + a_2 x_{n-2} + \ldots + a_p x_{n-p} \tag{4.2}$$

This is a linear form of relating the current sample with the previous ones. In addition to this, they can be related in a non-linear fashion for other forms. Such a relation is useful in relating the concurrent samples to form a suitable correlation structure.

- **I (Integrated)** This corresponds to the order of previous errors taken is consideration while adjusting for the current sample. For example, a d-th order integration will accumulate the errors from past d samples when matched against the empirical time series as:

$$e_n = e_{n-1} + e_{n-2} + \ldots + e_{n-d} \tag{4.3}$$

The errors accumulated thus help in matching the model with the traffic data. This way, after optimizing for the weights, a_i, the model can be made to match the traffic data.

- **MA (Moving Average)** To adjust the model to the traffic data it is essential to take into account the dynamic behavior of the time series. To do so, the average of the time series is adjusted according to the traffic data. The order of the moving average process corresponds to the number of previous samples considered in doing so.

Together an ARIMA process with p-th order autoregression, d-th order integrative and q-th order moving average process gives rise to an ARIMA(p, d, q) process. It is clear that ARIMA processes have more variables to optimize than the fGn process and hence yield greater flexibility in modeling the self-similar behavior. They have been shown to be very useful in modeling both long-range and short-term dependence. More specifically, ARIMA$(1, d, 0)$ and ARIMA$(0, d, 1)$ have been shown to be capable of capturing the self-similar behavior of most practical network traffic time-series.

ARIMA processes excel over others in terms of their greater flexibility and simplicity. The only problem with these processes is that it may be hard to optimize the weights to tune the model the empirical time-series. A suitable optimization algorithm needs to be chosen before actual optimization w. r. t. the actual traffic data is performed. Several optimization algorithms are available like:

- Quasi-Newton

- Genetic Algorithm(GA)
- Levenberg-Marquardt
- Classical Newton method
- Huber Method etc.

Among these, the network modeler can choose a suitable method to fine-tune the weights according to the traffic data. However the usefulness of the model is dependent on the order of the process chosen, suitability of the optimization algorithm and, to an extent, the experience of the network designer. The problem with ARIMA processes comes in the form of too much flexibility that makes it difficult to optimize with too many variables. A too rigid model with fewer parameters may be incapable of capturing the real network's self-similar behavior. On the other hand, a too loose model, with too many parameters to optimize may be difficult to be tuned to the actual network traffic behavior.

ARIMA(p, q, d) processes have been shown to be second-order self-similar with self-similarity parameter, $d + 1/2$, as long as, $0 < d < 1/2$. In fact, ARIMA processes are generalizations of the broader Box-Jenkins model [25]. The latter one offer even more flexibility than ARIMA processes, but it is even more difficult to control the many parameters of the Box-Jenkins model. Also, a narrowed down version of ARIMA processes is the ARMA processes which do not have the integrative terms. These are also useful in modeling the short-term autocorrelation behavior. Overall, ARIMA processes are difficult to be optimized for the empirical data, once optimized, they can be used to generate data in a fast and useful manner. Many simulators are increasingly using the ARIMA models for self-similar traffic generation in network simulators.

4.4 Artificial Neural Networks

Artificial neural networks(ANNs) have been introduced in modeling, simulation and optimization [6, 7]. Several researchers have effectively used the universal approximation properties of ANNs to model the self-similarity and network traffic and time-series prediction. Among the popular ANN approaches, dynamic neural networks (DNNs), have been used to model the

transient behavior of network traffic. The structure, popularly used in dynamic modeling of the network traffic is shown in Fig. 4.1. Neural Networks have been shown to exhibit universal approximation, i. e. they can capture any kind of non-linear behavior provided sufficient number of neurons are there in the hidden layer. The training data for the neural network comes from empirical network traffic data. Based on the training data, the neural network can be made to learn the network traffic behavior.

The modeling methodology for the network traffic involves feeding the neural network with some finite number of previous number of samples to get the current sample. Specifically, for a q-th order dynamic neural network, the network is first fed with samples $x[k-q]]$ through $x[k-1]$ to obtain the k-th sample, $x[k]$. This is then matched with the k-th sample in the training data. Based on the error in the predicted value, the internal weights of the neural network are adjusted accordingly. In the next iteration of the training the samples from $x[k-q]$ through $x[k]$ are used to calculate the $k+1$-th sample, $x[k+1]$. The error between the predicted and empirical value is then used to adjust internal neural network weights. This way the neural network is trained for the entire training data.

The power of neural newtroks emanate from very powerful optimization algorithms like *backpropagation* can be used to tune the weights of the neural network for the empirical traffic data. Backpropagation is both fast and effective. It has been shown to offer significant advantages over other optimization algorithms and are widely used by neural network researchers across the world. An example of use of neural network in modeling the traffic behavior is shown in Fig. where the model is compared against the empirical traffic data. It can be seen in Fig. 4.2 that neural model fairly captures the actual behavior(This work was done by this author sometime back).

Neural networks can be trained very fast and are very useful in predicting network performance and queuing characteristics as well. However, the major problem with neural networks comes with their poor performance outside the trained range. Beyond the trained range, neural networks behave arbitrarily and are not very useful. Several approaches [18] have been proposed to enhance the performance of a neural network outside its trained range with

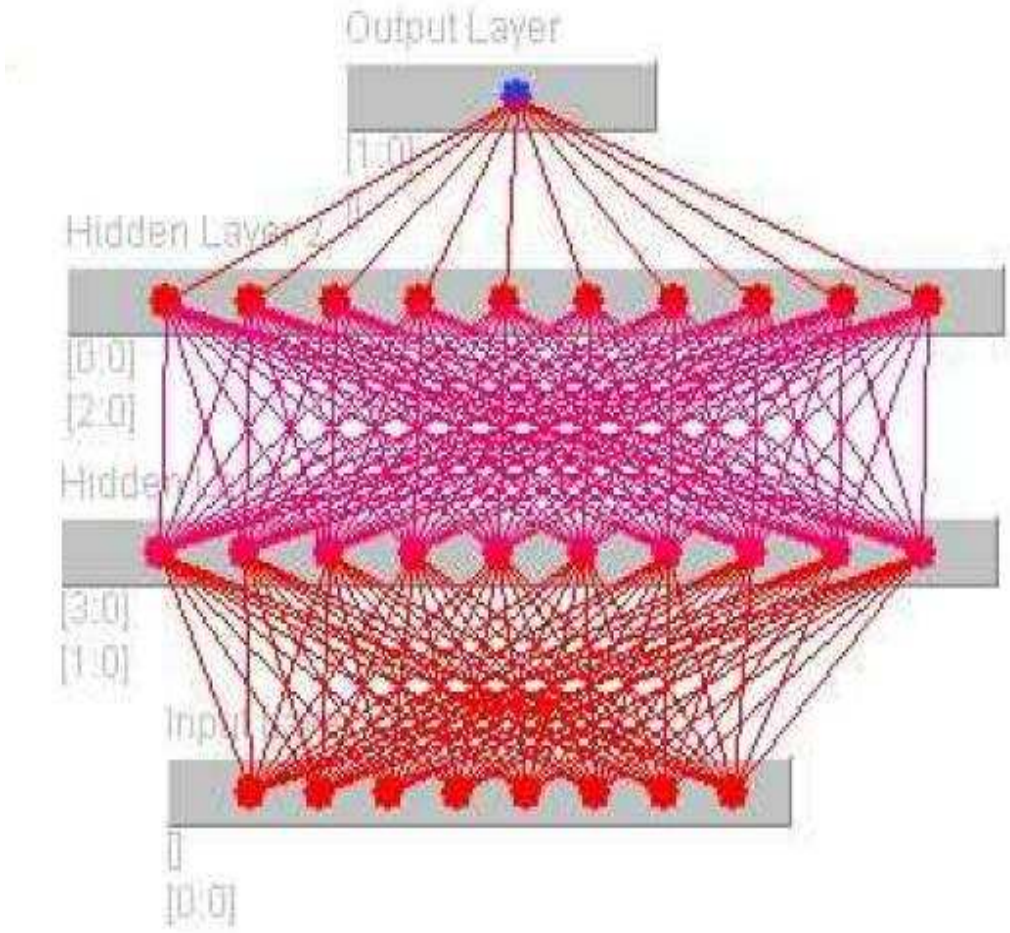


Figure 4.1: Dynamic Neural Network Structure

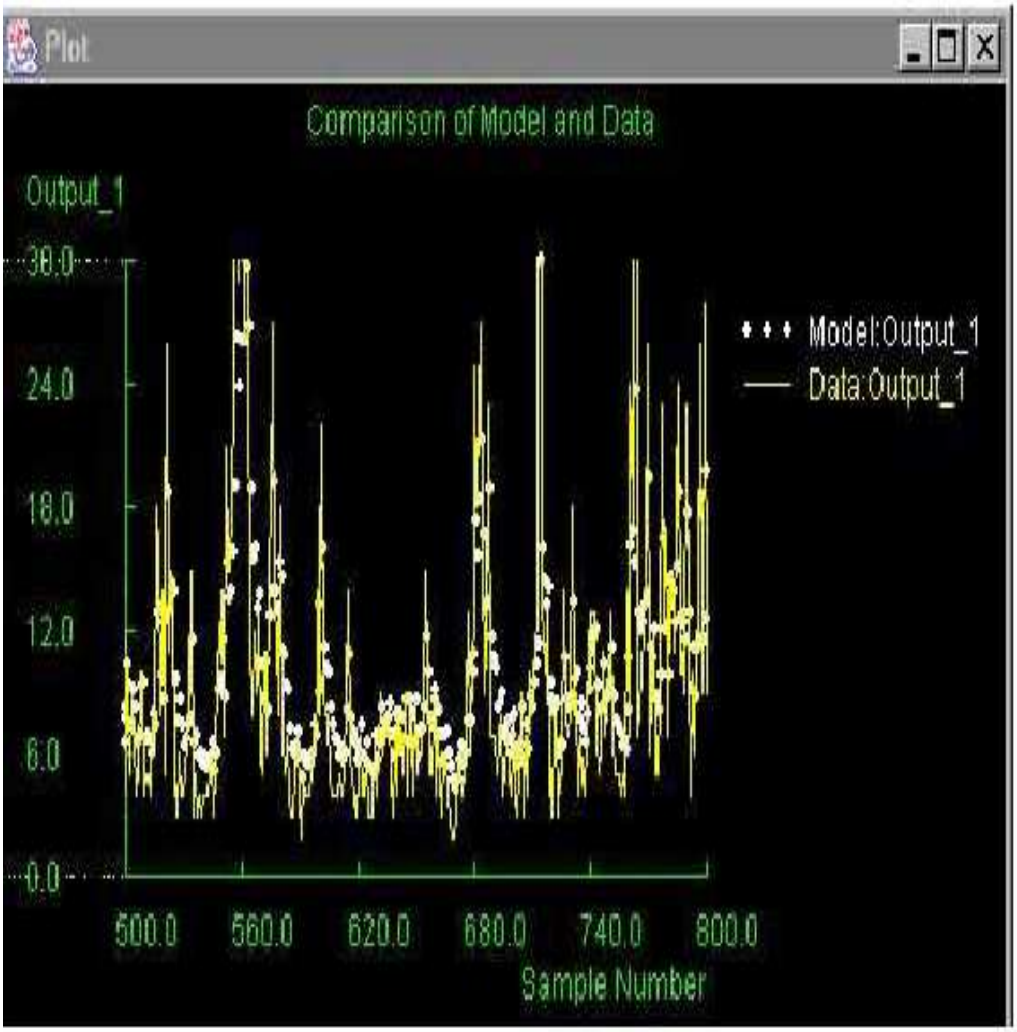


Figure 4.2: Comparison of Neural Model and original traffic data

limited enhancements. Nonetheless, neural networks are very useful in the sense of their ease of use and broad applicability.

On the front of capability in modeling the longer-term dependence, neural networks are not very useful beyond a point. The ability of a neural network structure to model long-term behavior is dependent on the order of the dynamic neural network structure, p. So, very long-range and sustained processes require higher order DNNs and hence need many neurons which make the whole network very cumbersome and complex. Also, the time complexity grows with the order of the neural network.

4.5 M/G/∞ queue-based method

In this method, a queue with Poisson arrivals, a long-range dependent service-time distribution and a infinite server system (i. e. pure delay system) is considered. We can look at each of them in detail:

- Poisson arrivals mean that the interarrival service-time distribution is exponentially distributed, i. e. a the arrival times are separated by a time that follows the distribution:

$$f_X(x) = Ae^{-\lambda x} \tag{4.4}$$

  where A and B are the locale and scale parameters respectively.

- A heavy-tailed service-time distribution is one which has infinite variance. With an infinite variance the service time can probabilistically attain any higher value of service time and causes heavy-tailedness. An example of a heavy-tailed distribution is:

$$F(x) = cx^{-\alpha}, \alpha > 0, x > \beta \tag{4.5}$$

- An infinite number of server means that an incoming packet need not wait in the queue for any time and the only latency in the systems comes from processing time of the packet.

With an M/G/∞ queue with components as shown above, the average queue length turns out to be:

$$queuelength = \frac{\lambda\beta\alpha}{\alpha - 1} \tag{4.6}$$

and the autocorrelation function is:

$$\tau(k) = \tau_0 k^{-(2-H)} \tag{4.7}$$

Such a autocorrelation structure is very useful in modeling the long-range dependence. This approach appears very natural in modeling the self-similar traffic behavior as it actually mimics the causes that are underneath the creation of self-similar traffic in real computer networks. The data is generated by sampling the queue length at a suitable sampling rate. The sampling rate is critical in accuracy. However the speed is impacted in a negative manner by the sampling rate. Hence a tradeoff has to be done in terms of sampling rate so as to attain higher levels of both speed and accuracy. The drawback of this method is is that the process is only asymptotically self-similar, so again one needs to make a tradeoff of length of computation and degree of self-similarity.

4.6 Transform-Expand-Sample(TES)

The Transform-Expand-Sample method was introduced in. This technique aims at matching the *pdf* and $autocorrelation function$ of the model and the empirical traffic daata and preserve the visual resemblance at the same time. For doing so, following methodology is used:

1. The histogram for the distribution function, $H(y)$ is developed based on the empirical data.

2. The distribution function is inverted to get the associated inverse, $H^{-1}(y)$.

3. A stitching transformation is then performed on the inverse function.

4. An innovation density with randomly chosen slots in the interval $[0, 1]$ is chosen and is finally applied to the inverted function.

TES methodology also involves optimizing the parameters for matching the correct behavior of network traffic. TES method, however, has the problem that it can hardly only short-term dependence and is not very good for long-range dependence. For longer runs, it involves optimization of many variables that makes it hard for any optimizing algorithm to get to a good solution.

For these reasons, TES method is suited only for short-range dependence. However the advantage for this technique comes from the fact that it already implemented in form of a useful software, TesTool that makes it an attractive choice for rough modeling of network traffic.

4.7 Comparative Evaluation

Based on the discussion above, the modeling techniques for self-similar traffic can be compared for various parameters as shown in the Table. 4.7. It can be seen that, overall, FGN and ARIMA processes come as a natural choice for the network designer to use as a model for generating self-similar traffic because of their broad coverage. Among the two, FGN excels over ARIMA because of its reported speed advantage over ARIMA with only exception that under certain circumstances, it may not be possible to use the FGN process as a modeling technique. However, it has been shown that for most practical purposes FGN generated network traffic is sufficient to model the self-similar behavior.

Property	**FGN**	**ARIMA**	**ANN**	**M/G/∞**	**TES**
Method	FFT	Linear Comb.	ANN	Sampling	TES
Theoretical Basis	Asympt. 2nd order	2nd order	Univ. Approx.	Asympt. s. s.	-
Analytical Tractability	$\surd$	$\surd$	$\times$	$\surd$	$\times$
Implementation Complexity	Very Simple	Moderate	Simple	Simple	Difficult
Optmization Parameters	μ, σ^2, H	Weights weights	Neuron	λ, β, α	weights
LRD support	Good	V. good	Limited	Good	Very Limited
S.S. Coverage	only a class	all covered	all covered	all covered	very limited
Speed	Fastest	Fast	Fast	Slow	Fast
Parameterized or Optmimized	Param.	Opt.	Opt.	Param.	Opt.

Table 4.1: Comparison of various modeling techniques for Self-Similar traffic

Chapter 5

Measurement Techniques

5.1 Introduction

To measure the efficacy of the modeling technique it is useful to measure the modeled traffic for various parameters. The parameters to be measured are :

- Sample mean, $\hat{\mu}$.
- Sample variance, $\hat{\sigma^2}$
- Hurst parameter, $\hat{H}$

First two of these can be measured using standard statistical tools while, for the last one, elaborate tools for measuring Hurst parameters are used. The estimation of H is somewhat more involving. The literature is full of variety of methods to measure H with their own advantages and disadvantages. Verification of the developed model is an essential part of the modeling methodology and needs to be done. Statistics has a major role to play in doing so. There are a variety of software tools offering statstical packages that can be used for the measurement purposes. In the following sections we will be looking at some of the popular techniques for measuring the self-similarity in modeled traffic data.

5.2 Variance-Time Plots

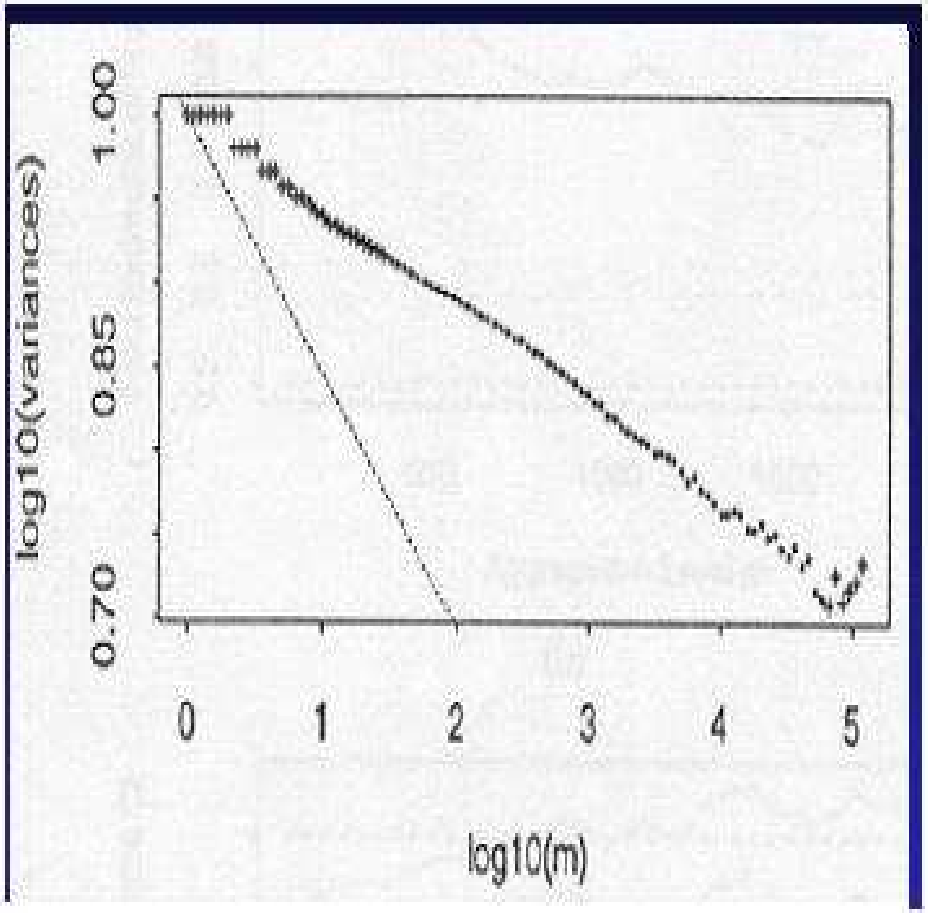


Figure 5.1: Variance-Time Plot (from [13])

Based on the definition of self-similarity in Section 2. 2, we can say that the aggregated processes, $\{X_k^{(m)}\}$ are strictly self-similar if,

$$var(X^{(m)}) = \sigma^2 m^{-\beta} \tag{5.1}$$

with a Hurst parameter, $H = 1 - \beta/2$. So this indicates that if we form the aggregated processes, $\{X^{(m)}\}$ for various values of m, and plot the logarithms of variances against the logarithms of m, then it will yield a straight line, the slope of which will give us the value of β. This is underlying idea behind measurement technique using the variance time plot[2]. It is a simple time-domain approach for measuring the Hurst parameter. In [13] it was shown using the variance time plots on empirical data that the ethernet traffic is self-similar. The same approach is used even today for a quick estimate of H. Fig. 5.1 shows one such V-T plot. The dotted line corresponds to self-similar traffic data whereas the solid line represents the hypothetical Poisson process.

5.3 R/S Analysis

For the estimation of Hurst parameter using R/S analysis, the following statistics is calculated first,

$$R(n)/S(n) = 1/S(n)[max(0, W_1, W_2, \ldots, W_n) - min(0, W_1, W_2, \ldots, W_n)]$$
$$W_k = (X_1 + X_2 \ldots + X_k) - k\ \bar{X}\ (n)(k \geq q)$$

where $S^2(n)$ is the sample variance and X_i are the samples from the traffic data. It has been shown that,

$$E[\frac{E(n)}{S(n)}] \sim an^H, n \to \infty \tag{5.2}$$

This property can be used in estimating the value of H[2]. First a logarithmic plot of $E[\frac{E(n)}{S(n)}]$ versus n can be drawn and then a regression analysis on the slope of the lines can yiled the value of H. Fig. 5.2 shows one such plot that is used to evaluate the value of H.

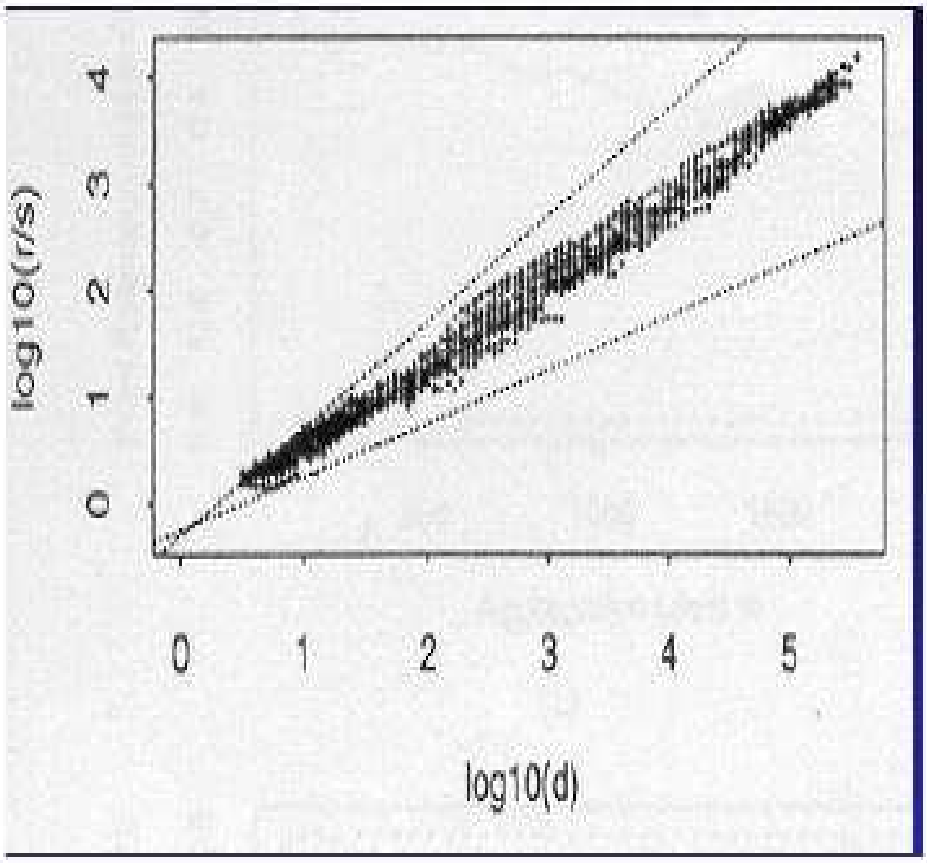


Figure 5.2: R/S Plot (from [13])

5.4 Wavelet Method

This method of estimation is based on discrete wavelet decomposition[24]. In doing so, first the coefficients of the discrete wavelet transform are calculated. This performed by partitioning the data into octaves and for each octave, an estimation of the long-range dependence parameter is made. It can be seen that LRD parameter, α and the Self-Similarity parameter H are related as

$$H = \frac{1+\alpha}{2} \tag{5.3}$$

An estimate for α is made in each of the octaves. Finally and average is being made out of the LRD parameters in different octaves and is output as estimate for α. This way the value of H can be calculated in the frequency domain. It can be noted that the previous two techniques are time-domain based, but wavelet method uses frequency domain for parameter estimation. It has been reported that this method offers significant advantages over other methods in terms of robustness, computational speed and in being an unbiased estimator.

Chapter 6

My Work

6.1 Aim

After having understood the self-similarity in network traffic, various modeling and measurement strategies, the aim of this project is to work on some issues related to self-similar traffic in Internet. Keeping in line with the idea behind this project, following things were planned to be done as part of this project:

1. To undertand Self-Similarity as a concept and its practical implications.
2. To do a review of some important modeling techniques for self-similar traffic and tools for measuring the self-similarity.
3. To choose one of the modeling approaches and develop a self-similar traffic model generator based on it.
4. To implement the approach in one of the available simulation softwares.
5. To verify the accuracy of the developed model using one of the measuring techniques.

In accomplishment of the final two goals, this chapter presents the implementation details for this project. It is desired in this process to conform to one of the latest simulation techniques and use them effectively.

6.2 Design Choices

For the purpose of implementation, following choices from the whole plethora of availbale tools and methods is made. Following are the choices that were made and the underlying logic behind them:

- **Methodology: M/G/∞ queue-based** As presented in section 4. 5, M/G/∞ is a simple and versatile method for fast generation of quality samples for self-similar traffic. The reason for making this choice lies in its inherent simplicity and usefulness in study of traffic data generation. This method involves closed form analytical formulae that make it a natural candidate for a network designer. All it requires in a simulator is a Random number generator and support for queues which are available in most of the discrete-event simulators. It is to be noted that this technique can be effectively extended to any other simulator as well.

- **Simulation Software : GPSS** General Purpose Simulation System (GPSS)[40] is a discrete-event simulator that is widely used for network and other practical simulations. (A discrete-event simulation is one in which the state of the system being simulated changes at only a discrete set of time points.) GPSS lends itself especially well to modeling systems in which discrete units of traffic compete with each other for the use of scarce resources, and is useful in determining how well such systems will respond to the demands placed on them. GPSS has been applied in diverse disciplines, medical, computer networks, telephone traffic, manufacturing systems etc. Overall, GPSS is a generic software for common queuing simulations. GPSS is a both simple and useful tool for the purpose of this project as well. Being a freeware, it can be easily downloaded and installed and simulations can be run to meet various kinds of requirements. GPSS has elaborate mechanisms for statistical analysis of queues that makes it an ideal choice for this work. M/G/∞ queues can be easily implemented in GPSS because of their queuing support. Due to minimal graphical overload, GPSS is fast and accurate in simulating large complex calculations.

- **Measurement Technique: Wavelet-based** Because of the reported advantages being offered by the wavelet technique as opposed to other time-domain methods, it is being used for verifying the accuracy of the

model. An implementation of this technique available in MATLAB is used.

6.3 Modeling Methodology

As has been described in section 4. 2, M/G/∞ method requires Poisson arrival, infinite servers and a heavy-tailed distribution. For Poisson arrivals, inter-arrival time in GPSS can be made exponential and for a heavy-tailed distribution Pareto distribution is used. GPSS supports queues with discrete arrivals and hence the desired setup can easily be done in it.

For the purpose of infinite number of servers, the available number of servers in made arbitrarily larger than average number of packets in the system. The service-time is the only factor causing delay in the processing time as the number of servers is such large that all incoming packets find an available server to get processed. This ensures that the stipulation of infinite number of servers is met with.

6.4 Implementation

The code shown in Fig. 6.1 is used to implement the desired queuing behavior in GPSS. The values of simulation parameters a prescribed in [15] is used to get the traffic data with desired self-similarity characteristics. GPSS has provisions for elaborate queuing analysis.

A description of the simulation script can be made like this. The keyword *STORAGE* creates an entity with specified number of capacity. This has been used to model an infinite number of servers by choosing an arbitrarily large number of servers. Variable definitions are made by the keyword *VARIABLE*. The keyword *QUEUE* creates a queue with name of it being specified. *ENTER* denotes the entry of an entity into the queue during the runtime and in a similar fashion, *DEPART* refers to departure of the entity. *GENERATE* command generated queuing entities at intervals distributed according to the specified function and *ADVANCE* is used to create the service time for the entity which is distributed according to a Pareto distribution as has been discussed in earlier sections.

```
* Author : Vikas Paliwal
* Create Servers, assigned an arbitrarily
* large number of servers
        InfServer STORAGE 65000
* Define Variables
        ArrRate VARIABLE 1.0
        Locale VARIABLE 10
        Scale VARIABLE 1.05
* Simulation
        GENERATE  (Exponential(1,0,V$ArrRate))
                        ;Create next arrival.
        QUEUE     ServerQ
                        ;Begin queue time.
        ENTER     InfServer
                        ;Take one of the server
        DEPART    ServerQ
                        ;End queue time.
        ADVANCE   (Pareto(2,V$Locale,V$Scale))
                        ;Service Time
        LEAVE     InfServer
                        ;Release the server
        TERMINATE
        GENERATE 0.01
                        ;Ratio for sampling
        TERMINATE 1
```

Figure 6.1: GPSS simulation script

After the end of simulations, GPSS creates a report mentioning the results of the simulation. This reprot is used in statistical analysis of the developed traffic model. The developed report for a sample simulation is shown in Fig. 6.2.

6.5 Verification and Results

A visual look at the generated traffic using this technique shows that visual self-similarity is apparent as seen in Fig. 6.3. It is clear that whatever be the time-scale, the traffic appear visullay self-similar. This illustrates that data generated using M/G/∞ technique shows self-similarity. More rigorous tests are also performed to verify its accuarcy.

As mentioned earlier the available implementation of verification tool in MATLAB is used. The traffic data generated using GPSS is fed to MATLAB and the distribution, autocorrelation and Hurst parameter are verified against the design targets. The MATLAB plots for obtaining the values using OCTAVE approach is shown in Fig. 6.4 and Fig. 6.5.

As mentioned earlier the design is tested for various values of H and the results are shown in Table. 6.5. The value of β is kept constant at 2 throughout.

6.6 Conclusion

Based on the verification process, it can be said that M/G/∞ is a useful modeling approach and is shown to offer a reasonably high level of accuracy. The only problem it runs with is lack of analytical tractability. A direct formula relating the design parameters α, β, λ would have been useful. In absence of a formula an optimization routine can be run to get the desired values. For our purposes, we have taken the values as prescribed in earlier works.

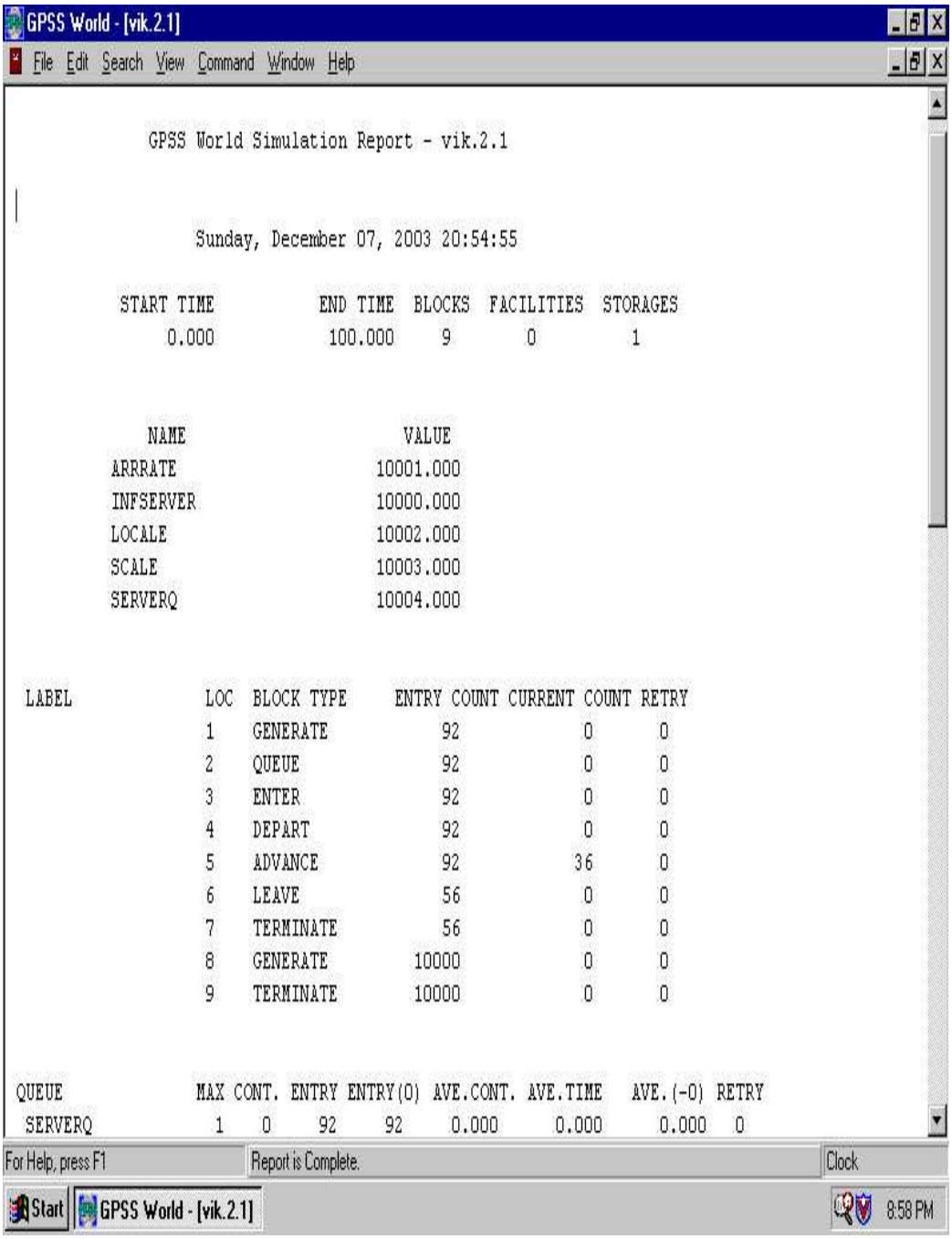
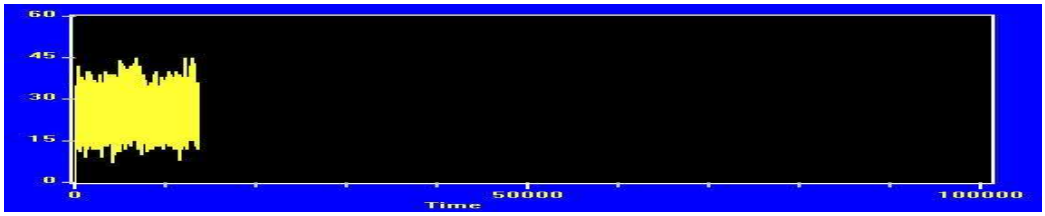
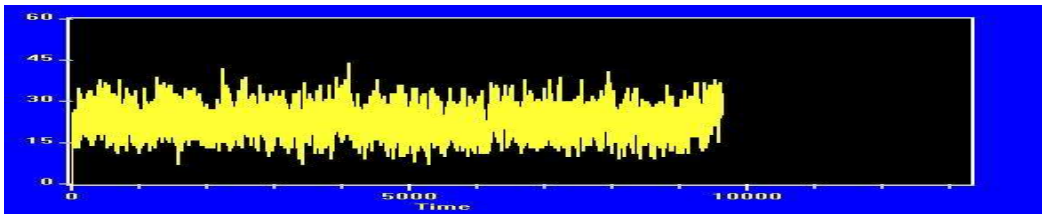


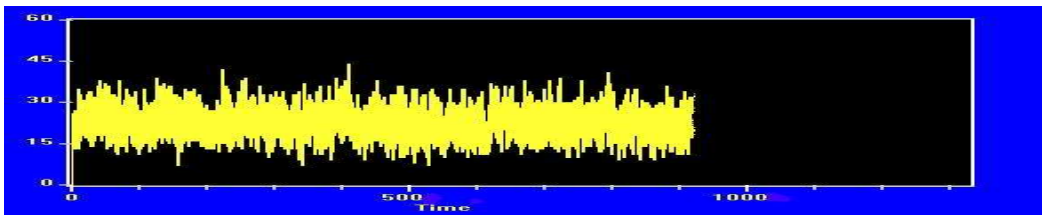
Figure 6.2: GPSS Report for one example simulation



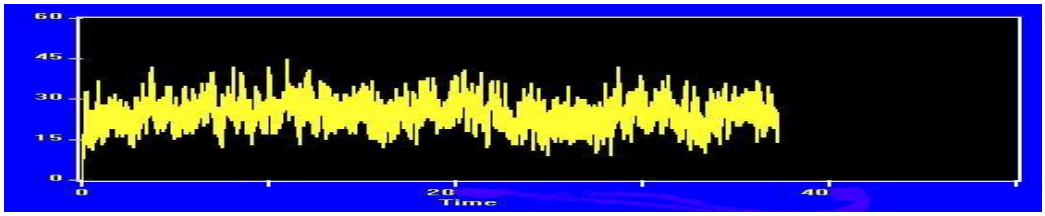
(a) scale=100s



(b) scale=10s



(c) scale=1s



(d) scale=0.04s

Figure 6.3: Traffic data generated seen at various time-scales

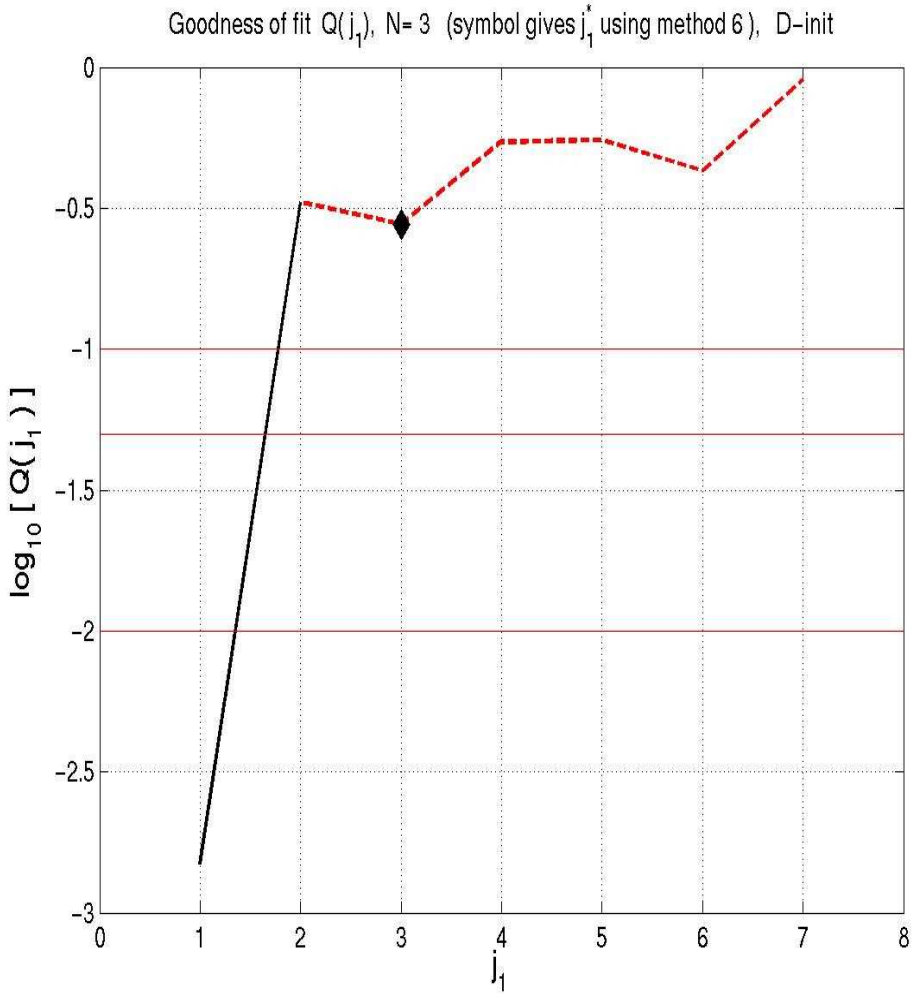


Figure 6.4: Octave splitting of traffic data

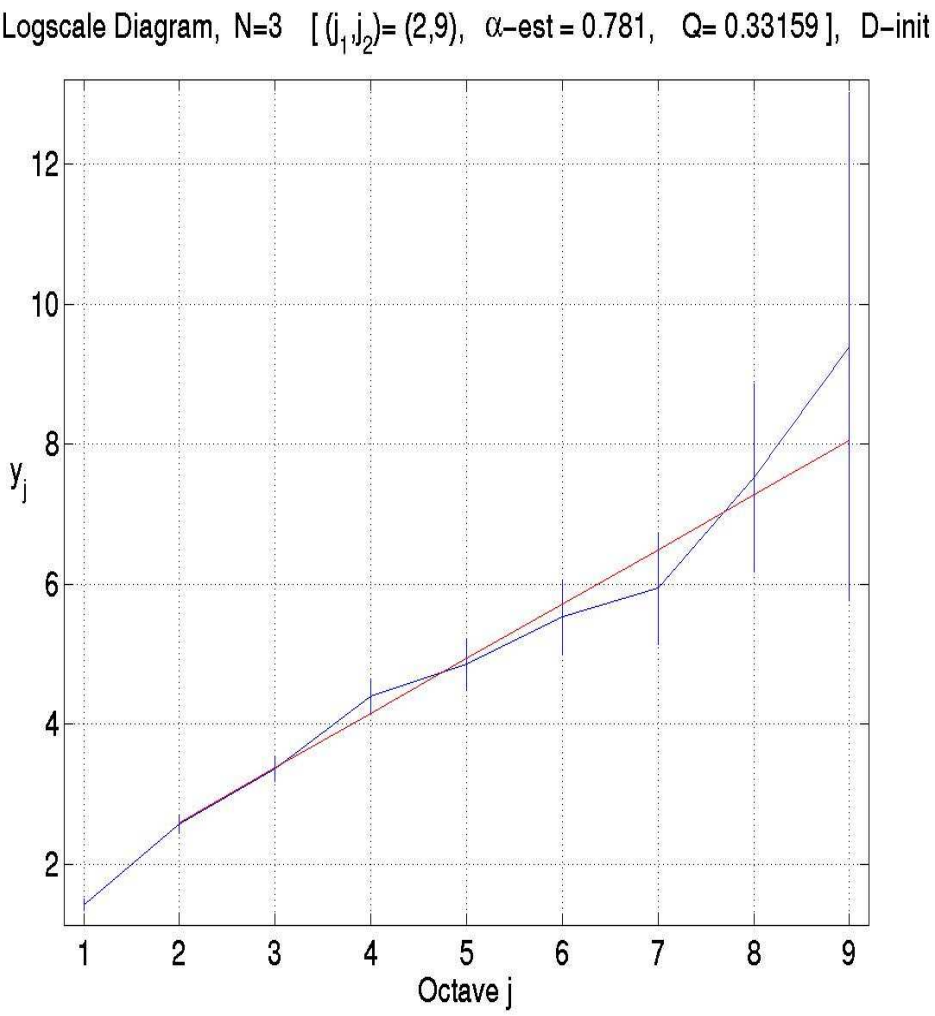


Figure 6.5: Estimation of LRD parameter using various octaves

S.No.	H_{spec}	λ	α	H_{model}
1	0.95	10	1.9	0.948
2	0.9	5	1.732	0.912
3	0.85	5	1.654	0.848
4	0.80	5	1.523	0.799
5	0.75	5	1.412	0.742
6	0.70	5	1.339	0.712
7	0.65	5	1.285	0.651
8	0.60	5	1.221	0.602
9	0.57	2	1.118	0.576
10	0.55	1	1.117	0.539

Table 6.1: Comparison of desired and measured value of H for different setups

Chapter 7

Current Research

7.1 Current Direction

The need for self-similar traffic modeling is now recognized more than before.[1] We can look at the ongoing research in diverse directions for modeling the self-similar traffic. But it is important to mention here that most of the current work is focussing on primarily FGN and ARIMA processes. Neural Networks have also been recently introduced for this purpose.

ARIMA process are being touted as the solution for modeling self-similar traffic in most of the existing network simulators. A recent modification to the classical ARIMA process [3] reports enhanced performance and accuracy and is reported to be better than the best known method on FGN side. the shift is currently going in favour of approximate methods as opposed to exact methods because of their computational speed. The goal for the future is to develop a broad modeling mechanism that covers whole range of self-similar processes. The research is getting more organized as issues like quality of traces generated using various modeling techniques are now being tested on common standards of autocorrelation structures. Previously it used to be such that methods with limited correlation structures like TES were thought to be mostly adequate (or it was thought a little modification would suffice) but now increasingly researchers are realizing the need for a real long-term generator for self-similar traffic.

On the other side research community is also working towards the direc-

[1] ACM is even organizing a major event on the completion of ten years of emergence of self-simlarity, see [1]

tion of analyzing the impact of self-similarity on network traffic. It began with application layer[23] but now even transport layer protocols like TCP are being studied in light of self-similar traffic [5]. On the prediction side, traditional methods like least mean [4], together with neural networks are being used for network performance analysis.

Overall the interest in this stream of traffic modeling is increasing with time and there is still a greater scope for even more research in this direction. Still the million-dollar question of, "What self-similar process must I choose?", looms at large over the minda of network designers and researchers across the world. It appears that ARIMA and FGN processes are the strong contenders for the universal modeling technology for self-similar traffic. However, there is still a need for substantial improvement in both these methods to cater the wide range of needs of a network designer.

7.2 Available Implementations

Several implementations covering some aspects of self-similar modeling are available:

- A popular version of FGN implementation based on FFT is available at [42].
- The Network Simulator [38] uses Pareto traffic generator and hence is a very useful tool in extensive evaluation of impact of self-similarity on network performance.
- A tool for R/S analysis at [36]
- A wavelet based estimator for measuring the self-similarity is available at [41]
- A Neural Modeler that can be used is available at [39].
- The GPSS software that is used in this work is at [40].
- A tool for Variance-Time Analysis at [37]

In addition to these, MATLAB has routines that can be used for self-similar traffic generation.

Chapter 8

Conclusion

In this work a formalism for critical study of various methods for self-similar traffic generation were studied. In doing so, efforts were made to understand self-similarity as a concept. Then an overview of various methods for modeling the self-similar traffic were studied. Also a comparative study of the methods was done from a designer's perspective. Also some of the measurement techniques were also studied. All this helped in developing a good understanding of the self-similarity in networks.

In addition to it, as a case study and implementation, one of the modeling techniques was chosen and implemented in a simulation software. Through various measurement techniques, the accuracy of the implementation was verified. It was seen that final model worked as expected albeit with little inaccuracy. Overall, the modeling technique was implemented and verified using suitable techniques.

In a nutshell, in this work, self-similarity was studied both in a global and a specific manner in the sense an overview of existing techniques was done in general and, in particular, M/G/∞ queue-based method was studied in detail.

Bibliography

[1] *http://www.acm.org/sigcomm/sigcomm2003/tutorial2.html*

[2] K. Park, W. Willinger,"Self-Similar Network Traffic and Performance Evaluation" , John Wiley, New York, 2000.

[3] J. Ardao, C. Garcia, "On the Use of Self-Similar Processes in Network Simulation",in *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. (10), pp. 125-151, April 2000.

[4] H. Zhao, N. Ansari, Y. Shi," Self-similar Traffic Prediction Using Least Mean Kurtosis ", *International Conference on Information Technology: Computers and Communications*, April 28 - 30, 2003 Las Vegas, Nevada.

[5] G. He, Y. Gao, J. Hao, K.Park "A case for exploiting self-simlarity of Network Traffic in TCP Congestion Control",in *Proceedings of the 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, 2002 pp. 34-43.

[6] H. Yosefizadeh"A Neural-Based Technique for Estimating Self-Similar Traffic Average Queuing Delay" *IEEE Communications Letters*, October 2002.

[7] H. Yosefizadeh, "Neural Network Estimation of Packet Arrival Rate in Self-Similar Queuing Systems" *Invited Paper, In Proceedings of the Second Workshop on Fractals, Power Laws, and Other Next Generation Data Mining Tools; The 9-th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2003.*

[8] S.Ostring, H.Sirisena"The Influence of Long-range Dependence on Traffic Prediction", *Proceedings of ICC'01*, Helsinki, June 2001.

[9] T. Tuan, K. Park,"Multiple Time Scale Congestion Control for Self-similar Network Traffic", *Performance Evaluation* 36-37(1-4), pp. 359-386, 1999.

[10] W. Willinger, M. Taqqu, D. Wilson, "Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level", *IEEE/ACM Transactions on Networking (TON)*, pp. 71-86, vol. 5, Feb.1997.

[11] Mark E. Crovella, Azer Bestavros,"Self-Similarity in World Wide Web Traffic Evidence and Possible Causes", *IEEE/ACM Transactions on Networking*, vol.2, 1996.

[12] J. Beran,"Statistics for Long-Memory Processes", *Monographs on Statistics and Applied Probability*. Chapman and Hall, New York, NY, 1994.

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

[14] V. Paxson, S. Floyd,"Wide-Area Traffic: The Failure of Poisson Modeling", in *IEEE/ACM Transactions on Networking*, vol. 3 (1995), pp. 1-18.

[15] A. Erramilli, P. Pruthi "Fast and Physically-Based Generation of Self-Similar Network Traffic with Applications to ATM Performance Evaluation", in *Proc. of 1997 Winter Simulation Conference*, pp. 997-1004.

[16] D. Cox"Long-Range Dependence : A Review", in *Statistics: An Appraisal*, IA, 1984, pp. 55-74.

[17] H.Fowler, W. Leland,"Local area network traffic charateristics, with implications for broadband network congestion management", *IEEE J. Selected Areas in Communications*, vol. 9, pp. 1139-1149, 1991.

[18] V. Paliwal, Q. Zhang"Circuit Modeling using Extrapolated Neural Networks", *IEEE-APM Conf.*,Seoul, S. Korea, 2003.

[19] B. Mandelbrot"Selfs-similar error clusters in communication systems and the concept of conditional staionarity", *IEEE Trans. Comm. Tech.*, vol. COM-13, pp. 71-90,1965.

[20] D.Veitch"Novel Models of Broadband Traffic", in *Proc. 7th Autralian teletraffic research seminar*, Murray river, Australia, 1992.

[21] C. Huang, M. Devetsikiotis, I. Lambadaris, A. Kaye,"Fast Simulation for Self-Similar Traffic in ATM Networks" *IEEE ICC '95* , Seattle, Washington, June 18 - 22, 1995.

[22] V. Paxson,"Fast, Approximate Synthesis of Fractional Gaussian Noise for Generating Self-Similar Network Traffic", in *Computer Communication Review*, vol. 5 (1999), pp. 14-128.

[23] K. Park, G. Kim, M. Crovella,"On the Effect of Traffic Self-Similarity on Network Performance", *IEEE/ACM Transactions on Netwroking*, vol. 1(2001), pp. 234-256.

[24] D. Veitch, P. Abry,"A Wavelet Based Joint Estimator of the Parameters of Long-Range Dependence", in *Journal of Statistics*, vol. 2 (2000), pp. 134-181.

[25] G. Box, G. Jenkins *Time Series Analysis : Forecasting and Control*, 2nd ed. , San Francisco, CA: Holden Day, 1976.

[26] C. Huang, I.Lambadaris, A. Kaye"Modeling and Simulation of self-similar variable bit-rate compressed video: A unified approach" in *Proc.of ACM SIGCOMM'95* Cambridge, MA, pp. 114-125.

[27] B. K. Ryu, A. Elwalid,"The Importance of Long-range Dependence of VBR Video Traffic in ATM Traffic Engineering: Myths and realities", *ACM Computer Communication Review*, vol. 26, pp. 3-14, Oct. 1996.

[28] J. Peha,"Retransmission mechanisms and self-similar traffic models", *IEEE/ACM/SCS Communications Networks and Distributed Systems Modeling and Simulation Conference*, Phoenix, Arizona, pp. 47-52, Jan. 1997.

[29] M. Grossglauser, J. Bolot,"On the Relevance of Long Range Dependence in Network Traffic" *IEEE/ACM Transactions on Networking*, 1998.

[30] B. Ryu, S. Lowen,"Point-Process Approaches to the Modeling and Analysis of Self-Similar Traffic", in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Fransisco, California), Mar. 1996.

[31] A. Feldmann, A. Gilbert,W. Willinger,"Data networks as cascades: Investigating the multifractal nature of Internet WAN traffic", *ACM Computer Communication Review*, vol. 28, pp. 42-55, Sept. 1998.

[32] N. Duffield,"Applications of Large Deviations to Performance Analysis with Long-range Dependent traffic", Keynote talk for *Workshop on Stochastic modeling and analysis of communication networks*, Lund, Sweden, October 1-3, 1998.

[33] N. Duffield,"On the relevance of long-tailed durations for the statistical multiplexing of large aggregations" in *Proceedings of the 34th Annual Allerton Conference on Communication, Control and Computation*, October 2-4 (1996).

[34] N.Duffield,"Predicting Quality of Service for traffic with long-range fluctuations",in *Proceedings of IEEE International Conference on Communications*, Seattle, 18-22 June, 1995, pp473-477.

[35] A. Feldmann, A. Gilbert, W. Willinger, T. Kurtz,"The Changing Nature of Network Traffic: Scaling Phenomena" , *ACM Computer Communication Review*, vol. 28, pp. 5-29, Apr. 1998.

[36] *ftp://ftp.ics.uci.edu/pub/duke/self-similar/*

[37] *http://www.ics.uci.edu/ atm/self-similar.html*

[38] *http://www.isi.edu/nsnam/ns/*

[39] *http://www.doe.carleton.ca/ qjz*

[40] *http://www.minutemansoftware.com/FTPdownload.htm*

[41] *http://www.cubinlab.ee.mu.oz.au/ darryl/secondorder_code.html*

[42] *http://www.winlab.rutgers.edu/s3/reu/week2.html*