ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL and COMPUTER ENGINEERING

Internet Self-Similarity, Modelling and Performance Evaluation

By
Taye Abdulkadir

January 2003

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL and COMPUTER ENGINEERING

Internet Self-Similarity, Modelling and Performance Evaluation

By
Taye Abdulkadir

Advisor
Prof. G. Devarajan

A thesis submitted to the School of Graduate Studies of Addis Ababa University in partial fulfilment of the requirements for the Degree of Masters of Science in Electrical Engineering

Declaration

I, the undersigned, hereby declare that this thesis is my original work, has not been presented for Degree in any other University and that all sources used for the thesis have been duly acknowledged.

Taye Abdulkadir Edris

Addis Ababa University.

Acknowledgments

I would like to thank my advisor, Prof. G. Devarajan, for helping me in every way an advisor can be at help. I also thank ECE Department Chairman Dr. Ing Mohammed Abdo for his support. My appreciation also goes to my cousin Mustafa Idris for providing me with the necessary books from abroad, which cannot be found in the country. Thanks also to Mesfin Ayalew for his friendly advises and support. I would also like to thank colleagues and friends at Icon Networks and Ethiopian Telecommunication Corporation for their support during packet trapping. My deep appreciation also goes to my Parents for their patience and support. Last, but not least, I would like to extend my appreciation to those who where there when I needed them until this work is finalized.

Table of Contents

List of Abbreviations

ARIMA	Auto Regressive Integrated Moving Average
ARMA	Auto Regressive Moving Average
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Networks
CB-WFQ	Class Based-Weighted Fair Queueing
FARIMA	Fractional Auto Regressive Integrated Moving Average
FBM	Fractional Brownian Motion
FGN	Fractional Gaussian Noise
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
ISDN	Integrated Services Digital Networks
LAN	Local Area Networks
LFI	Link Fragmentation and Interleaving
LLQ	Low Latency Queueing
LRD	Long Range Dependence
MPLS	Multi Protocol Label Switching
POTS	Plain Old Telephone Systems
PQ	Priority Queueing
PSTN	Public Switched Telephone Networks
QoS	Quality of Service
RTO	Retransmission Time Out
RTP	Real Time Protocol
RTT	Round Trip Time
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol
WAN	Wide Area Networks
WFQ	Weighted Fair Queueing

ABSTRACT

Understanding the *nature of traffic* and its *associated delay process* in high traffic systems, such as the Internet, is essential for Engineering, Operations & Performance evaluation of these networks. It has been a common practice to assume packet arrivals as a Poisson Process in modelling high Traffic Networks. However, data communication traffic levels fluctuate over time, and delays through congestion can occur even on lightly utilized links. These fluctuations can occur over very short periods of time giving rise to the concept of a burst of traffic. This high variability or bursty nature can be explained through *Self-Similarity and Long-Range Dependence*. In this thesis work an attempt is made to verify the Self-Similarity or Long-Range Dependence nature of *Internet Traffic* and its associated *Packet Round Trip Delay* process. Also Fractional Auto-Regressive Integrated Moving Average (FARIMA) Model is found to fit the long memory as well as the short memory properties of the collected Internet Traffic Data. An artificial traffic trace synthesis program based on the FARIMA model is also developed.

While the research convincingly establishes the presence of LRD over a wide range of time scales in the packet traffic and delay processes, its implication to Queueing performance is a center of argument, in one hand arguments acknowledging LRD in packet traffic affecting Queueing performance, and in the other hand arguments emphasizing LRD has no practical impact and need not be incorporated into performance models. In this research it is tried to do experimentation, which demonstrates the first argument, that LRD in traffic indeed affects Queueing Performance.

TCP, the reliable transport protocol of TCP/IP, employs an *ACK* (*acknowledgment*)-based window control, that is, a *TCP sender* updates its congestion window size every time it *randomly* measures Round Trip Time (RTT) or Packet Round Trip Delay. With the most basic mechanism in the Internet to detect losses, the sender *re-transmit* a segment if it's ACK has not been received in the expected amount of time. Therefore, TCP *throughput performance* depends on the RTT. Investigation on the relation between RTT Self-Similarity and TCP throughput performance as measured by the amount of bandwidth wastage due to premature timeouts is also covered.

Real time traffic must get a fairly regulated number of packets through to the destination in a timely manner. If these packets are late enough to have missed their “play” window, they are useless. Voice over Internet Protocol (VoIP) is one real-time application carried over the Internet. Packet delay variations in Internet tend to create voice degradations, unless proper Quality of Service (QoS) measures are implemented. In this thesis work investigation on Packet Delay and VoIP relations have been done and QoS measures suggested.

Chapter 1

Introduction

Leland, et al [1] present a preliminary statistical analysis of the fractal nature of a High quality *Ethernet* data collected from Bellcore Morristown Research and Engineering Center and commented in detail the presence of "Burstiness" across an extremely wide range of time scales. This fractal like behaviour of aggregated Ethernet LAN Traffic is different from conventional telephone traffic, and different researches afterwards argue that other high traffic networks, such as the *Internet,* also have similar fractal like behavior.

Burst appears in the counting process of traffic, as packets tend to come in clusters. The burst phenomenon cannot be smoothed out by simply aggregating the traffic in a *larger time scale.* That is, even using very large time units to construct the counting process, the *burst* phenomenon can still be observed. This demonstrates that the underlying process of measured traffic is *Self-Similar*. Intuitively, self-similar phenomena display structural (and/or statistical) similarities across all (or at least a very wide range of) time scales.

Self-similarity and *fractals*, notions pioneered by Benoit B. Mandelbrot [2], describe the phenomenon where a *certain* property of an object for example, *a natural image, the convergent sub domain of certain dynamical systems, a time series* (the *mathematical* object of our interest)- is *preserved* with respect to *scaling* in space and/or time. If an object is Self-similar or fractal, its parts when magnified resemble -in the suitable sense - the shape of whole.

Stochastic Self-similarity admits the infusion of non-determinationism as necessitated by measured traffic traces but, nonetheless, is the property that can be illustrated visually. *Unlike deterministic fractals*, the objects do *not* possess exact resemblance of their part with the whole at finer details. Indeed, for measured traffic traces, it would be too much to expect to observe exact, determinist self-similarity given the *stochastic* nature of many network events (e.g. source arrival behaviour) that collectively influence actual network traffic. Second order statistics are statistical properties that *capture* burstiness or variability, and the *Autocorrelation* function is a yardstick with respect to which scale invariance can be fruitfully defined.

This nature of *time-invariant burstiness* is completely different from the traditional *tele-traffic models,* such as the *Poisson Process,* which have a "smoothed-out" burst structure as the time aggregation increases; that is, after a certain time scale there no surprises due to spikes of Traffic.

Despite the facts that *data networks* such as the Internet are drastically different from legacy public switched telephone networks, the long held paradigm in the Communication and Networking research community has been that *data traffic -analogous to voice traffic* - is adequately described by certain *Markorian* Models which are amenable to accurate analysis and efficient control. This supposition has been instrumental in shaping the optimism permeating the late 1980's and early 1990's regarding the ability of achieving efficient traffic control for quality of service provisioning in modern high-speed communication networks. The discovery and, more importantly, successive formulation and recognition that actual data traffic may, in fact, be fundamentally *different* in nature from the hereto accustomed telephony traffic has significantly influenced the networking research landscape, necessitating a re-examination and revamping of some of its basic premises.

Therefore, any Data traffic Network should be modeled differently than the usual Tele Traffic Models. In recent years, empirical studies on network traffic both in local area and wide area networks argue that the properties of actual traffic are very different from that predicted by Traditional Tele Traffic Models, such as the Poisson process. For actually measured traffic the correlation in traffic can extend to a wide range of different time, or mathematically, the correlation function of realistic traffic decays with lag time in the way of *power law*, which is the property of the so called *Long Range Dependence (LRD)*; while for traditional models of generated traffic, its correlation function decays exponentially fast; namely *Short Range Dependence*. One of the goals of this research work is to assert this argument in one high traffic network-the Internet.

The importance of scaling in traffic is clear as far as *modeling* of the traffic itself is concerned, that a feature as prominent as scaling should be built into models at a fundamental level, if these are to be both accurate and parsimonious. Scaling, therefore, has immediate implication for the choice of traffic models, and consequently on the choice, and subsequent estimation of model parameter. Such estimation is required for initial model verification, for fitting purpose, as well as for traffic monitoring. Fitting of a proper *Mathematical Model* that catches the scale invariant nature as best as possible and *synthesis of artificial traffic* based on this model are considered in this research.

Understanding the nature of traffic cannot be a final goal by itself. Networks carrying traffic should have their *performance metrics* properly tuned. *Packet Round Trip Delay* is an important metric of network performance. Quantifying delay is essential in building proper *Quality of Service (QoS)* provisioning for the next generation of multimedia services. The *Statistical* property of Packet Round Trip Delays should be studied. The fractal nature of the packet traffic is *expected* to dominate performance measures of the Communication Network. In this research work the statistical nature of the Packet Round Trip Delay process is investigated. It is proved that *Scale-invariance or Self-Similarity* is also present in the Packet Round Trip Delay process.

Knowledge of the impact of Traffic Self-Similarity on *Queueing Performance* is crucial in quantifying delays, since Queueing delay is the most important constituent of packet delay. The approaches in studying Queueing performance centered around two arguments. Some

argue that LRD has its impact on Queueing performance. Others argue that LRD has no practical impact on Queueing performance, since Queueing performance is determined by features in arrival processes of the time series of a Queueing system's busy period. In this research work, Investigation of the effect of traffic self-similarity on Queueing performance is also included. With simulation tools it is asserted contrary to the second argument, and indeed LRD of packet traffic has an effect on Queueing performance. Therefore, ways to curtail this effect of LRD must be implemented at the intermediate Queues along packet path, which assure acceptable Quality of Service.

The nature of the Packet Round Trip Delay process has its own impact on transportation mechanisms that implement Round Trip Times (RTT) for their proper functioning. One of these transportation mechanisms used in the Internet is TCP, an *ACK (acknowledge)* based communication protocol of the TCP/IP protocol stack. TCP, therefore, has to be *tuned* properly using measured RTT values. Failure in tuning will result in *wastage of bandwidth* due to premature time outs, which result in packet re-transmission. Investigation of the effect of Packet Round Trip Self-Similarity on the performance of TCP has been carried out. The result showed that more premature timeouts occur as the Packet Round Trip Process become more self-similar.

Delay causes problems in *Real-Time or Constant Bit Rate (CBR)* packets. Data or Variable Bit Rate (VBR) packets affected more by packet loss than packet delays. Real-time traffic must get a fairly regulated number of packets through to the destination in a timely manner. If these packets are late enough to have missed their "play" window they are useless. In this research work, investigation on the effects of packet delays on one Internet real-time application-VoIP is discussed. Quality of Service issues to counteract these effects are covered.

The following sections give brief ideas and concepts supporting the analysis procedures to be done throughout the research. Further discussions on the Internet and VoIP are covered in Chapter 2 and 3, respectively. Mathematical tools used in the research are discussed in Chapter 4.

Internet Traffic Characterstics

The Internet has experienced a fascinating evolution, especially since the early days of the web. *Unprecented* in its growth, *unparalleled* in its heterogeneity, and *unpredictable* or even *chaotic* in the behaviour of its Traffic, "the Internet is its own evolution". Chapter 2 has more on the Internet.

The term "Traffic Theory" originally encompassed all mathematics applicable to the design, control and management of the Public Switched Telephone Networks (PSTN); Statistical

Inference, Mathematical Modeling, Optimization, Queuing and Performance Analysis. Later, its practitioners would extend this to include data networks such as the Internet.

Traditional tele traffic theory as applied to POTS (Plain Old Telephone Service) has arguably been one of the most successful applications of mathematical technique in industry. [6] It has led to first-rate telephone networks, quality of service we fully rely on and take for granted. Among the main reasons for this tremendous success of tele-traffic theory and practice in traditional telephony are the highly *static* nature of conventional PSTNs and a well defined and ever present notion of limited variability, a trade mark of homogenous systems where one talks about *"typical" users* and *"generic" behaviour* and where *averages* describe the system performance adequately.

The static nature of traditional PSTNs contributed to the popular belief in the existence of "universal laws" governing voice networks, the most significant of which is the *presumed Poisson* nature of call arrivals at links in the network traffic is heavily aggregated. This law states that call arrivals are mutually *independent*, and call inter arrival times are all *exponentially* distributed, with one and the same parameter.

Ironically, the complacency engaged by this mathematical elegance and (particularly) success has recently been rocked by changes in the "static" world of telephony. More than fifty-years of old models use, the bedrock of the Teletraffic modelling success story, now have been greatly undermined by two major "new" uses of the telephone network. Their changes began with the advent of FAXes in the 1980s and have continued and become more drastic with the popularity of the Internet.

The key change is that telephone calls used for FAX transmissions and Internet accesses have *statistical* characterstics drastically different from a typical call. Both types of calls are now playing havoc with the existing PSTN engineering infrastructure designed to deal with voice calls only.

One might expect that the voice network modelling success story would enjoy another triumph when applied to data networks, and indeed this has been attempted. But in fact much of the voice traffic modelling has proven nothing short of disastrous when applied to data networks; for the simple but profound reason that the rules all change when it is *computers* not humans doing the talking.

Voice traffic has the property that is relatively *homogenous* and *predictable*, and from a signaling perspective, spans long time scales. Consequently, many concurrent voice connections can be easily "Multiplexed" to share a common (expensive) wire or "link" by allocating a fixed amount of the link's capacity to each connection. Voice networks have been engineered in a *Circuit Switching* fashion.

In contrast to voice traffic, data traffic is much more *variable* with individual connections ranging from extremely short to extremely long and from extremely low rate to extremely high rate. These properties have led to a design for data networks in which each individual data "packet" or "datagram" transmitted over the network is forwarded through the network *independently* of previous packets that may have been transmitted by the same connection. Each packet is self contained and the routes need only inspect the "header" of the packet to determine its destination and forward it through the network. Consequently, the Routers do not keep track of each currently active connection.

This shift away from Circuit Switching toward Packet Switching has profound implication. On the one hand, it results in highly *efficient* networks. Anytime capacity is available in the network, newly arriving packets can benefit from it. Each packet in the network competes with all the others. If there happens to be little competing traffic along a particular path, then a connection using that path can enjoy the entire "Bandwidth" and transfer its data very quickly. If many connections compete along the same path, then each will receive a (perhaps unfair) portion of available bandwidth.

In addition, packet switching buys enormous *Robustness*: it enables networks to route *transparently* around Router or link failures without perturbing active connections. Routers have no problem accepting the re-routed traffic because, as far as they can tell, it is not in anyway "new" traffic - they have no notion of "current" traffic and hence no problem accepting traffic they did not until that very moment know existed -a situation very *different* from circuit-switched networks, of which routers cannot easily accept re-routed traffic because they have no knowledge of the corresponding virtual circuit.

However, links can become overloaded because packets arrive for transmission along them at rates exceeding the capacity of the link. Such packets will be buffered awaiting transmission along the link, but if the excess rate is sustained, - a condition termed "congestion"- then ultimately the buffers in the routes will fill up and some packets must be discarded, or dropped. To ensure the sources behave properly in the presence of congestion in the network, the *protocols* used for transmitting data include end-to-end congestion control mechanism that decrease automatically the rate at which data are transmitted when congestion is detected. An important consequence of the use of congestion control is that traffic in the Network is shaped by the conditions each connections has encountered in its past. Thus, Internet traffic includes a basic mechanism that introduces significant complicated *correlations* across time, as well as complex interactions among the active connections.

A damaging legacy of the telephony influence on data network research was a virtually complete absence in the 1970s and 1980s of attempts to validate crucial modelling assumptions against actual data network traffic measurements. Data traffic is highly variable or very "*bursty*". That it does not come at a steady state, but instead it starts and fits with lulls in between. The term "bursty" has a readily understood intuitive meaning, but it turns out that nailing down its precise, mathematical meaning has profound implications for developing *mathematical model* of Network Traffic.

The Natural approach for getting a handle on burstiness is to define interms of a *time scale* over which activities and lulls occur. For telephony this time scale is related to the *rate* of the Poisson Process that describe the dynamics of call arrival. Then inverting the rate gives the time scale. Chapter 4 presents proofs that imply traffic bursts in Internet do indeed occur on many *different time scales,* and that such multiscale burstiness simply does not fit the world of traditional Poisson based traffic modelling.

Figure 1.1 is a visual demonstration of the failure of Poisson modelling to capture the burstiness present in actual network traffic. The difference between the Poisson Model and the measured traffic is obvious and striking: as the time scale increases the Poisson Traffic "smoothes out", becoming quite tame, while the measured traffic shows no such predilection. The difference is crucial from an Engineering perspective: Traffic that behaves as shown in Figure 1.1(a) can be easily engineered for. Above a certain time scale there are no surprises – every thing boils down to knowing the long term arrival rate; no need for big buffers in routers or switches, no reason for being conservative in choosing safe operating points for engineering back bone trunks, and why even think of user perceived Quality of Service as being a relevant issue? In stark contrast, measured traffic like that showed in Figure 1.1(b) is "wild", remains so even on quite coarse time scales, and plays havoc with conventional traffic engineering: Routers require big buffers to accommodate the traffic fluctuations across many time scales; in the absence of any effective controls, safe operating points have to be set conservatively, because the traffic can saturate the link at any time and over any time scale; and adequate overall Network Performance can no longer be taken as a guarantee of satisfied individual users.

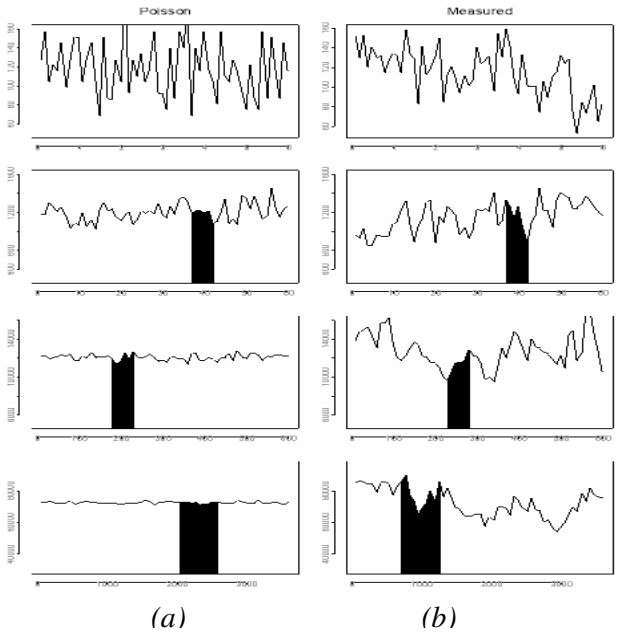


Fig 1.1 (a) Synthesized traffic from a Poisson model Vs. (b) Internet traffic to which its mean and variance fit, viewed over three orders of aggregation. Source: reference [6]

The edifice of Poisson Modelling repeatedly told us to expect the behavior shown on Figure 1.1(a) -but what is observed in realty is the roller coaster on Figure 1.1(b) The Internet engineering community has thus come to consider teletraffic theory as *irrelevant* to the development of the Internet.

Many networking experts argue that the only way to gain an in-depth understanding of data traffic is –simply put- doing away with teletraffic tradition and starting from scratch. Interestingly, *Mathematics*, which has been largely responsible for the success story of the teletraffic theory for the voice network, has recently provided strong ammunition in supporting of the Networking experts' arguments. However, as voice traffic turns out to differ drastically from data traffic, so do the underlying mathematical ideas and concepts.

The relevant mathematics for POTS is one of *limited variability* in both time (traffic processes are either independent or have temporal correlations that decay exponentially fast) and in space- i.e. the distribution of traffic related quantities have exponentially decaying tails. But for data networks, the mathematics is one of *high* or *extreme variability*.

Statistically, temporal high variability in traffic processes is captured by *Long Range Dependence*, i.e Autocorrelations that exhibit power law decay. On the other hand, extreme forms of spatial variability can be described parsimoniously using *heavy tailed distribution* with infinite variance, i.e, probability distribution F with the property that for large x values

$$1-F(x) \approx k_1 x^{-\alpha} \qquad (1.1)$$

Where k_1 is a positive finite constant that does not depend on x and where the tail index α is in the interval (0,2). (For example, this property is satisfied by the well-known family of "Pareto Distribution", originally introduced for modelling the distribution of *income within the population*.[3])

It turns out that Power Law behavior in time or space of some of their statistical descriptions often cause the corresponding traffic process to exhibit Fractal characterstics. In the present context we say that a traffic process has fractal characterstics, if there exists a relationship between certain *quantities* Q of the underlying process and *resolution* as the general form

$$Q(\tau) \approx k_2 \tau^{f(D)} \qquad (1.2)$$

Where k_2 is a positive finite constant that does not depend on τ. $f(B)$ is simple, often *linear*, function of D; D is a fractal dimension. To declare fractality, the above relationship is supported to hold for range of different τ-values with a value of D that is less than the embedded dimension.

In view of the general skeptism that exists in the different circles in the mathematical community concerning the needs, usefulness, and appropriateness of fractals, what can one say about fractal like scaling in measured date network traffic? To examine this question, we call a discrete time covariance stationary, zero mean *stochastic* process $X = (X_k : k \geq 1)$ *exactly* ***Self-Similar or Fractal*** with scaling parameter $H \in (0.5,1)$ if for all levels of aggregation or "Resolution" $m \geq 1$.

$$X^{(m)} =_d m^{H-1} X \qquad (1.3)$$

Where the equality is understood in the sense of Finite dimensional distributions, and where the aggregated process $X^{(m)}$ are defined by

$$X^{(m)}(k) = m^{-1}(X_{(m-1)k+1} + ... + X_{km}) \qquad (1.4)$$

When assessing the validity of describing a process using a *self-similar model*, one must be very careful not to mistake actual non-Stationarity for highly variable but stationary fractal behavior. The two can appear very similar, both to the eye and to a number of statistical tests. However this concern can be addressed by making good use of the *very large* size of Network traffic traces.

The switch from Poisson to Fractal thinking in Network traffic research has had a major impact on the understanding of actual network traffic to the point where we now know why aggregate Internet Traffic exhibits fractal-scaling behavior over different time scales. A measure of the success of this shift in thinking is that the corresponding mathematical arguments are at the same time rigorous and simple, are in full agreement with the Networking researchers' intuition, and can be explained readily to a non-networking expert.

Stochastic Self-Similar processes can be studied through Second Order Statistics. Second order statistics are statistical properties that capture burstiness or variability. The Autocorrelation function is a yardstick with respect to which scale invariance can be fruitfully defined. Unlike traditional models, such as the Poisson Process, Self-Similar processes have correlations that span longer time units. Due to this, the Autocorrelation function decays

slowly, as a power law, unlike the exponential decay expected from traditional models. This nature tells us the process has Long-memory or it is Long-Range dependent. Figure 1.2 presents visualization on the nature of the Autocorrelation functions of Long-memory and Short-memory processes. Chapter 4 presents a detailed discussion on the *mathematics* behind Self-Similarity and Long-Range dependence.

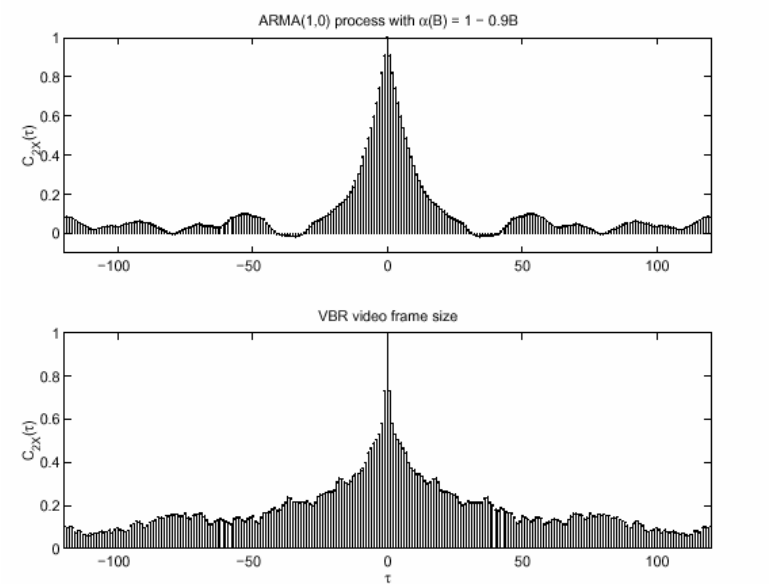


Fig1. 2: Autocorrelation of time series (a) Short Memory and (b)

Modeling Network Traffic

High-speed Network traffic mechanisms are propelling the growth of a new class of communication services such as Multimedia, Video on Demand, etc. As these new communication services evolve and the needs of the users change, existing communication backbone needs to be modified or replaced by completely newer systems. Telecommunication professionals are being called upon to design and manage these systems in the face of fast-moving technology and a climate of increasing customer expectations. Towards this end, B-

ISDN networks now attempt to provide a guarantee on the quality of service (QoS) deliverable to the customers.

Design and management decisions are mainly based on *models* that predicate the performance of Network. Analytical techniques, computer simulation, projections from existing experience and experimentation are methods used to evaluate and compare network design and protocols.[9]

Performance models require *simulation* of traffic using models that can accurately capture the statistical characteristics of the actual traffic. If the traffic models do not accurately represent the actual traffic, then one may overestimate or underestimate network performance.

Fractional Autoregressive Integrated Moving Average or FARIMA model proved to be a better choice for modeling High-speed communication networks, such as LAN and WAN, since it offers flexibility in modelling both short-memory and long-memory characteristics. Once the model is selected, synthesized artificial traffic based on that model is used in Performance Analysis and other related areas.

Chapter 5 presents a detailed explanation of the *candidate* mathematical models to represent the actual traffic data, namely ARMA, ARIMA and FARIMA. By fitting the periodogram of the collected traffic data to the spectrum of the fitted mathematical model, FARIMA processes proved to represent Internet Traffic. Presentation of synthesis procedures based on Hosking's algorithm, as applied to the FARIMA model, is also included.

Self Similarity and Network Performance

The studies made in this thesis work regarding the collected Internet Traffic data confirm the presence of scale invariance in different time scales. So the *performance* of Network should be dominated by this property. As a major performance metric of the Internet, further

investigation on the nature of Packet Round Trip Delays will give an insight to the implementation of measures to boost Network Performance.

Packet Round Trip Delay process when displayed as a time series also proved to have the scale invariance bursty nature or self-similarity. This finding is essential in building proper *Quality of Service (QoS)* provisioning for the next generation of delay sensitive real-time multimedia traffic to be carried over the Internet.

Once the existence of self-similarity is asserted in both the packet traffic and delay processes, focus was made on the influence of *LRD in traffic and delay* processes on the performance of Networking, which is approached from *three* directions. One is investigation of the effect of Traffic Self-similarity on Queueing performance. The second is investigation on the effect of Packet Round Trip Delay Self-Similarity on the performance of Internet transport mechanisms. As a third approach, performance of *Real-Time* traffic, in this case VoIP, investigated in relation with the nature of Packet Delays.

Queuing Performance with LRD packet traffic is seriously degraded. Numerical and analytical studies based on models catching LRD property of Traffic demonstrate that the tail of a queue length distribution decays much more slowly than the exponential rate; this implies that a packet tends to experience *longer* delay with in networks on average before it gets to its destination than that predicated by traditional models; such as the Markov Model. The conclusions drawn from these studies have remarkable implications for the design, control of networks, and tuning of protocols.

Different transport mechanisms are employed in the Internet. One of the commonly used transport mechanisms is TCP, included in TCP/IP, which is the protocol suite used in the Internet. With TCP there are several *time out parameters* needed to be carefully tuned, the most sensitive one is the Retransmission Time Out (RTO) parameter determined dynamically from packet round trip delays in Internet. The RTO is used to indicate when a packet can be

assumed lost in the Network and consequently invoke a retransmission event. Failure in tuning of RTO will result in *wastage of bandwidth* due to premature time-outs, which result in packet re-transmission. With the high variability nature Packet Round Trip Delay process, the Retransmission Time Out (RTO) parameter get highly affected and bandwidth wastage due to unnecessary re-transmissions is expected.

Chapter 6 presents detailed statistical analysis and experimentation on Packet Round Trip Delays. It includes investigation on the cause of Self-similarity of the Packet Round Trip Delay process. Queueing performance analysis and TCP performance analysis in relation with self-similarity are also investigated.

Real time or *Constant Bit Rate packets* travel through the Internet are the ones that are most affected by the packet delays encountered in the Internet or by the overall performance. Voice over IP (VolP) uses the Internet protocol (IP) to transmit voice as packet over an IP network. So VoIP can be achieved on any data network that uses IP, like Internet and Local Area Networks, by digitizing the voice signal, converted to IP packets and then transmitted over the IP network. Signaling protocols are used to setup and tear down calls, carry information required to locate users and negotiate capabilities.

For VoIP to become popular and in the long run to replace PSTN, Quality of Service (QoS) issues need to be resolved. As IP was designed for carrying data, it does not provide real time guarantees but only provides *best effort* services. VoIP will be successful if one gets best understanding of Packet Delays in the IP network. So for voice communication over IP to become acceptable to users, the delay needs to be minimized to an acceptable value. The capability to provide *resource assurance* and *service differentiation* in a network is often referred to as quality of service (QoS). Resource assurance is critical for many new Internet applications to flourish and prosper. The Internet will become a truly multiservice network only when service differentiation can be supported. Implementing these QoS capabilities in the Internet has been one of the toughest challenges in its evolution, touching on almost all aspects of Internet technologies and requiring changes to the basic architecture of the Internet.

Quality of service in a Multi-service network depends essentially on two factors; the *service model* that identifies different service classes and specifies how network resources are shared, and the *traffic engineering* procedures used to determine the capacity of these resources.

The *traditional* role of traffic engineering is to ensure that a telecommunication network has just enough capacity to meet expected demand with adequate Quality of Service. The critical

requirement is, therefore, to understand the three-way relationship between demand, capacity and Network performance, each of these being quantified in appropriate units.

The inherent Self-Similar nature resulted in high variation and long-range correlations of the Internet add more difficulty in QoS provisioning. Increasing network recourses alone will not be a solution for better Quality of Service, unless its tradeoff effects against self-similar traffic properly understood. Investigation on the relation between Internet packet delay metrics and VoIP requirements are included in Chapter 6. The Quality of Service mechanisms to increase voice quality are derived from these investigations. Chapter 3 discusses working principles of VoIP, its relation with Packet Delays and QoS measures to improve voice quality.

Chapter 2

The Internet

2.1 Introduction

First, the word *internet (*also *internetwork)* is simply a contraction of the phrase *interconnected network.* However, when written as a capital "I" the *Internet* refers to a *worldwide* set of interconnected networks, so the Internet is an internet, but the reverse does not follow. One of the greatest things about the Internet is that nobody really owns it. It is a global collection of networks, both big and small. These networks connect together in many different ways to form the single entity that we know as the ***Internet***.

Since its beginning in 1969, the Internet has grown from four host computer systems to tens of millions. However, just because nobody owns the Internet, it doesn't mean that it is not monitored and maintained in different ways. The *Internet Society*, a non-profit group established in 1992, oversees the formation of the policies and protocols that define how we use and interact with the Internet.

Every computer that is connected to the Internet is part of a network. For example, one may use a modem and dial a local number to connect to an *Internet Service Provider* (ISP). At work, the user may be part of a local area network (LAN), but the LAN most likely still connect to the Internet using an ISP that the company has contracted with. The ISP may then connect to a larger network and become part of that network. Therefore, the Internet is simply a network of networks.

Most large communications companies have their own dedicated ***backbones*** connecting various regions. In each region, the company has a ***Point of Presence*** (POP). The POP is a place for local users to access the company's network, often through a local phone number or dedicated line. The amazing thing here is that there is no overall controlling network. Instead, there are several high-level networks connecting to each other through ***Network Access Points*** or NAPs. This is shown in Figure 2.1.

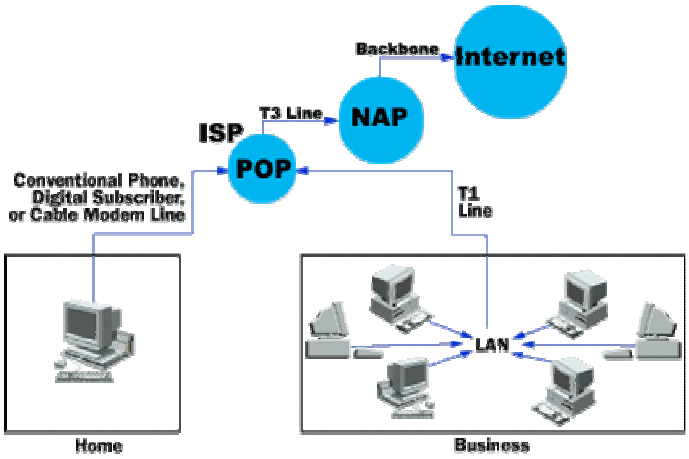


Fig. 2.1 Parts of the Internet

All of these networks rely on *NAPs*, *backbones* and *routers* to talk to each other. What is incredible about this process is that a message can leave one computer and travel halfway across the world through several different networks and arrive at another computer in a fraction of second.

Backbones are typically *fiber optic* trunk lines, even though the first backbone to get introduced by the *National Science Foundation* (NSF) was a T1 line operated at 1.544 Mbps. The trunk line has multiple fiber optic cables combined together to increase the capacity. Fiber optic cables are designated *OC* for O*ptical Carrier*, such as OC-3, OC-12 or OC-48. An OC-3 line is capable of transmitting 155 Mbps while an OC-48 can transmit 2,488 Mbps (2.488 Gbps). Compare that to a typical 56K modem transmitting 56,000 bps and you see just how fast a modern backbone is.

Today there are many companies that operate their own high-capacity backbones, and all of them interconnect at various NAPs around the world. In this way, everyone on the Internet, no matter where they are and what company they use, is able to talk to everyone else on the planet. The entire Internet is a gigantic, sprawling agreement between companies to intercommunicate freely.

2.2 Routing and the Internet

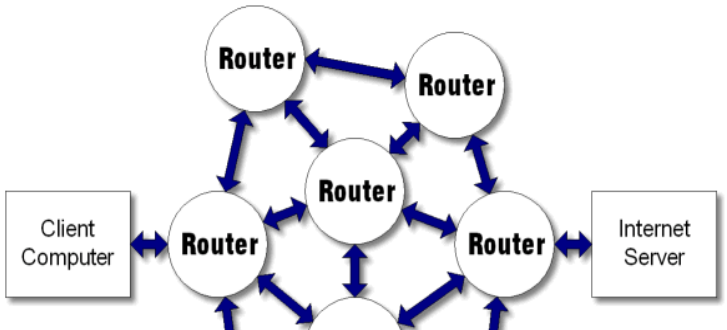
Current data networks typically use packet switching as a means of dynamically allocating network resources on a demand basis. Packet switching had been widely used because it facilitates the *interconnection* of networks with different architectures, and it provides flexible *resource allocation* and good *reliability against node and link failure.* Packets of a single traffic stream may take different routes and reach the intended destination. This process of assigning packets to available routes is called *Routing*. And the devices facilitating this operation are called *Routers*.

The *Routers* determine where to send information from one computer to another. Routers are specialized *computers* or *dedicated devices* that send packets speeding to their destinations along thousands of pathways. A Router has two separate, but related, jobs:

- It ensures that information doesn't go where it's not needed. This is crucial for keeping large volumes of data from clogging the connections of "innocent bystanders."
- It makes sure that information does make it to the intended destination.

In performing these two jobs, a Router is extremely useful in dealing with two separate computer networks. It joins the two networks, passing information from one to the other. It also protects the networks from one another, *preventing* the traffic on one from unnecessarily spilling over to the other. Regardless of how many networks are attached, the basic operation and function of the router remains the same. Since the Internet is one huge network made up of tens of thousands of smaller networks, its use of routers is an absolute necessity.

As can be seen in Fig 2.2, data moves between the two endpoint machines by "hopping" from one Internet router to the next and traveling across the many connecting links in between. Different packets of a single connection may take different routes.



Routing of the Internet has two features: *Flexibility and Scalability.* The Internet provides the *Dynamic* Routing based on the exchange of the *Routing Information* among Routers. For example, when a Network link becomes *down* because of some trouble, an alternative route will be prepared automatically. And also the packet processing at the routers is simple *(E.g. First In First Out(FIFO))* to reduce the overhead of packet forwarding at the router, except when special Quality of Service provisioning techniques employed.

2.3 TCP/IP and the Internet

Each Internet communication consists of a *transfer* of information from one computer to another; examples are the downloading of a Web page and the sending of an email message. When a file is transferred, it is not sent across the Internet as a continuous block of bits. Rather the file is broken up into pieces called *packet*s, and each packet is sent individually. Many different *protocols* collectively carry out the transfer. A protocol is simply a set of rules for communication between computers. The two core protocols are TCP, the *Transmission Control Protocol* and IP, the *Internet Protocol*.

TCP runs on both computers. It breaks up a file to be transferred into packets, sends them out from the source headed for the destination, receives them at the destination, and reassembles them into their proper order. A typical packet size is 1460 bytes. [15]

TCP does the transfer by establishing a *connection* between the computers. The connection is not a physical path; rather, it is simply TCP software executing on the computers in a coordinated fashion, with each aware that it is working with the other. The connection continues until both sides agree that it is over, or until one side fails to hear from the other for a specified amount of time. Packets go back and forth between the two computers. For a packet, the *source* computer is the one that sends it out, the *destination* computer is the one that receives it.

IP is in charge of *routing TCP's packets* across the Internet. Each computer has an IP address, a ***unique*** 32-bit number. The number is displayed by dividing the sequence of bits into four 8-

bit fields, called *Octets*. The number of possible IP addresses is 2^{32}, which is about 4 billion, but due to growth of the Internet, plans have been made to switch to 128 bit addresses, called *IP version 6*. The octets are used to create *classes* of IP addresses that can be assigned to a particular business, government or other entity based on size and need. The octets are split into two sections: *Net* and *Host*. The Net section *always* contains the first octet. It is used to identify the network that a computer belongs to. Host (sometimes referred to as Node) identifies the actual computer on the network. The Host section always contains the last octet.

When the Internet was in its infancy, it consisted of a small number of computers hooked together with modems and telephone lines. You could only make connections by providing the IP address of the computer you wanted to establish a link with. This was fine when there were only a few hosts out there, but it became unwieldy as more and more systems came online.

The first solution to the problem was a simple text file maintained by the Network Information Center that *mapped names to IP addresses*. Soon this text file became so large it was too cumbersome to manage. In 1983, the University of Wisconsin created the *Domain Name System* (DNS), which maps text names to IP addresses automatically.

An *IP header* is added to each packet. The header includes, among other things, the source IP address, the destination IP address, and the packet size. *Routers* get the packet to its destination. The source, the series of routers, and the destination form a path across the Internet. The packets of a single file transfer do not have to follow the same path; for example, if a path in use at the beginning of the transfer becomes unavailable, tables are updated by Communication among the Routers, and a new path is used.

A TCP header, which contains information to control the connection and to reassemble packets, is also added to each packet. TCP also creates packets that contain only headers but no file information. Control is their only purpose and they are generated by both computers. The control information in headers plays a role in a TCP feedback mechanism that is fundamental to the operation of the Internet and has a major effect on traffic. At the onset of a connection, TCP sends packets at a conservative rate and then progressively increases the rate. If a router receives packets faster than it can forward them, it places the overflow in a buffer; if the buffer overflows, packets will get dropped. When packets are dropped, TCP decreases the transmission rate, retransmits the dropped packets, and then progressively increases the rate again. *Fig 2.3 shows the TCP/IP protocol stack.*

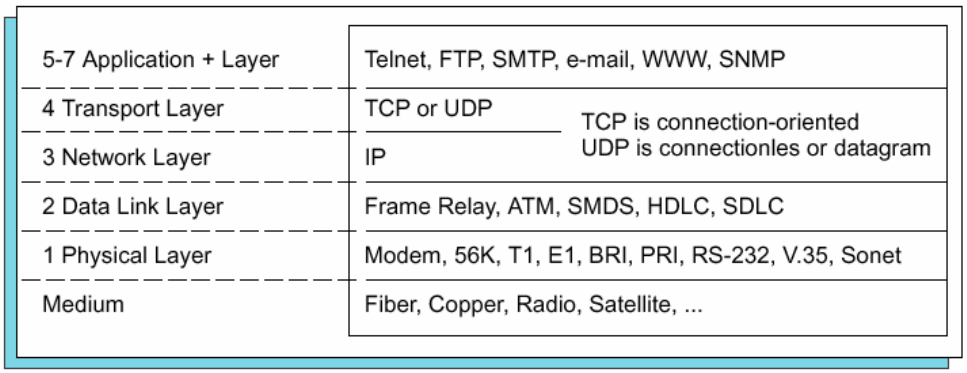


Fig 2.3 The TCP/IP protocol stack.

Another commonly used transportation mechanism included in the TCP/IP protocol stack is User Datagram Protocol (UDP). *UDP*, like TCP, functions to send data so that it arrives at the correct destination and in good condition. UDP, however, unlike TCP does not cause the originating device to *retransmit* the data if an error is detected. And UDP *does not* first "handshake" with the receiving end, allowing for faster initiation of the sending process — e-mail, for example, rides on top of UDP, unlike file transfer (FTP), which rides on top of TCP.

TCP provide a reliable transport layer. One of the ways it provides reliability is for each end to acknowledge the data it receives from the other end. But data segments and acknowledgments get lost. TCP handles this by setting a *Retransmission Timeout (RTO)* when it sends data, and if the data is not acknowledged before the time out expires, it retransmits the data. A critical element of any implementation is the time out and retransmission strategy. How the timeout interval is determined, and how frequently a retransmission occurs, greatly impact in the *utilization of the available bandwidth.*

Typical implementations measure the *Round Trip Time* of TCP segments and use these measurements to estimate the retransmission timeout. Round Trip Time can vary over time, as routes and network traffic change, thus TCP should track these changes and modify its timeout accordingly. TCP must recalculate its Retransmission Time Out by taking randomly measured RTT values. Investigation on how RTT self-similarity affects RTO calculations is presented in Chapter 6.

The TCP/IP protocol stack has a number of applications: to mention some SMTP (Simple Mail Transfer Protocol), Telnet, FTP (File Transfer protocol), Ping, Traceroute etc. Ping and Traceroute are discussed, as they are related to packet delay measurements in the Internet.

2.3.1 PING

Ping is the simplest of all TCP/IP applications. The word *ping,* which is used as a noun and a verb, is taken from the sonar operation to locate an underwater object. It is also an abbreviation for *Packet InterNet Groper.*

It tests whether another host is reachable. The program sends an ICMP (*Internet Control Message Protocol)* echo request message to a host, expecting an ICMP echo reply to be returned. Ping measures the *Round Trip Time* to the host, giving us some indication of "how far away" the host is. Ping is able to calculate the Round Trip Time by storing the time at which it sends the echo request in the data portion of the ICMP message. When the reply is returned, it subtracts this value from the current time.

Traditionally, if you could ping a host other applications like Telnet or FTP could reach that host. With the advent of *security measures* on the Internet, particularly *Firewalls,* which control access to Networks by application protocol and/or port number, this is no longer strictly true. Nonetheless the first test of reachability for a host is still to attempt to ping it. Most TCP/IP implementations support the *Ping Server* in the kernel, that is, the server is not a user process.

In this research work *Ping* is used extensively to measure the Round Trip Times or Delays. The information collected from continuous *Pings* by a custom application developed in Visual Basic is taken as a time series and its behavior studied.

2.3. 2 TRACEROUTE

The Traceroute program is a handy *debugging* tool that allows us to further explore the TCP/IP protocols. Although there are no guarantees that two consecutive IP datagrams from the same source to the same destination follow the same route, most of the time they do. Traceroute lets us see the *route* that IP datagrams follow from one host to another

Traceroute is based upon ICMP. It sends an IP datagram with a TTL *(Time To Live)* of 1 to the destination host. The purpose of the TTL field is to prevent datagrams from ending up in infinite loops, which can occur during congestion. The first Router to see the datagram will decrement the TTL to 0 and return an ICMP Time Exceeded message as well as discarding the datagram. In this way the, the first router in the path is identified. This process can be repeated with successively larger TTL values in order to identify the series of Routers in the path to the destination host. Traceroute actually sends UDP datagrams to the destination host which reference a port number that is outside the normally used range. This enables Traceroute to determine when the destination host has been reached, that is, when an *ICMP Port Unreachable* message is received.

2.4 Characterstics of Internet Traffic

Internet engineering and management depend on an understanding of the characteristics of network traffic. *Statistical models,* which can generate traffic that mimics closely the observed behavior on live Internet wires, are needed. Models can be used on their own for some tasks and combined with network simulators for others. But the challenge of model development is immense. Internet traffic data are ferocious. Their statistical properties are complex, databases are very large, Internet network topology is vast, and the engineering mechanism is intricate and introduces feedback into the traffic.

Packet header collection and organization of the headers into connection flows yields data rich in information about traffic characteristics and serves as an excellent framework for modeling. Many existing statistical tools and models, especially those for time series, point processes, and marked point process, can be used to describe and model the statistical characteristics, taking into account the structure of the Internet, but failed to exhaustively represent the properties of the Internet traffic's inherent nature. Therefore, new tools and models are needed. Internet traffic data are exciting because they measure an intricate, fast-growing network connecting up the world, transforming culture, politics, and business. And a deep analysis of Internet traffic can contribute substantially to network performance-monitoring, equipment planning, quality of service, security, and the engineering of Internet communications technology. Analysis can be made on traffic measurements to produce statistical models.

TCP connection flows provide a large amount of information. Each flow is an end-to-end connection traversing the Internet. The TCP/IP headers contain the IP addresses of the two computers, so one can know their location in the vast Internet topology. Thus flows can be used to study network-wide characteristics.

A TCP connection flow database also provides information about the traffic on the wire. The TCP/IP headers have the size of each packet in bytes, so together with the timestamps, one can have the aggregated packet process: the arrival times and sizes of all packets. Studying aggregates is important because the devices at each end of a wire must handle packets, in time order, and the performance of the devices depends on the packet inter-arrival times and the packet sizes. Forming the aggregate of all packets from the flows takes us back to the packet information in its original state: packets in time order. But storage by connection flow is still important because we often study sub-aggregate traffic: time-ordered packets from a subset of the flows.

Internet traffic data are ferocious. Their statistical properties are complex and databases are very large. The protocols are complex and introduce feedback into the traffic system. Added to this is the vastness of the Internet network topology. This nature of the Internet challenges modeling and analysis. Most Internet traffic data can be thought of as time data: a point process, a marked point process, or a time series. The start times of TCP connection flows for HTTP on an Internet wire are a point process. If we add to each of these start times the file size downloaded from the server to the client, the result is a marked point process. Byte counts of aggregate traffic summed over equally spaced intervals are a time series, as implemented in this research work.

Modeling Internet traffic data will require new approaches, new tools, and new models for time series data. Long-range Dependence is pervasive in Internet traffic data. But the pervasiveness had to be discovered. After the discovery of long-range dependence, Internet traffic can be studied through the vehicle of *Self-Similar* processes, invoking the creative work of Mandelbrot. Traffic models for voice traffic, developed over the years to serve the telephone network, did not apply as might have been hoped because voice traffic does not give rise to the same traffic characteristics as Internet data traffic, which is *burstier*.

2.5 Contents of Internet Traffic

Voice, video, and data traffic have *traditionally* been carried on separate networks. Much of the separation is due to the history of each technology's evolution and the different uses of these fundamental services. Today, however, people want to communicate simply and effectively, and there are many applications with which voice, video, and data can converge onto the same wire to provide meaningful enhancements to the communication experience. But what type of wire should it be? Is it better for data to travel on an otherwise voice circuit? Can data and voice coexist? These debates have raged since the first modem shipped, but the debate now appears to be over. In converged applications, it is commonly acknowledged that it is better for voice and video to run on a data infrastructure.

There are several compelling reasons for this consensus. First, voice and video are now actually digital data. Standards have paved the way for putting voice and video onto Internet Protocol (IP) networks, like the Internet, and service providers have long been evolving their networks in this direction. Second, voice, video, and data must truly intermingle in order to take full advantage of convergence. This can only be accomplished on high-performance data networks because even advanced *voice networks*, such as Integrated Services Digital Network (ISDN), keep voice and data traffic segregated in separate channels. Separate data streams cannot add value to each other, and therefore limit the overall user experience. Finally, IP networks are on a steep slope of innovation that will make them the long-term awaited carriers of all traffic types.

The convergence in communication gives rise to two innovative applications to be carried over IP networks, Voice over IP and Video over IP. Voice and video have very different characteristics and are always handled in different packet flows. Both voice and video can operate either in real-time (for example, Internet Telephony or Video Conferencing) or non-real-time (for example, voice mail or video playback).

A detailed discussion on VoIP is presented in Chapter 3. Video has gone through a transformation similar to that of voice. Codecs convert video from cameras into compressed digital streams, but there are more alternatives from which to choose.

Video virtually never travels without audio. The alternative audio standards must therefore be added to the basic video rates. Since the video and audio operate over separate logical channels and are processed by separate codecs, they can get out of synch. To prevent this from impairing the user experience, the Real Time Protocol (RTP) and Real Time Control Protocol (RTCP) maintain the *pairing* of the audio and video signals.

Video transmission over a packet network can also be greatly simplified by buffering at the destination. Such video streaming is popular on the Internet. The overall data rate does not change for the quality of video and audio desired. The quality can be maintained, however, over a loaded network that does employ guarantees of bandwidth or priority.

2.6 Packet Delays and the Internet

A packet round-trip delay is the sum of delays on each subnet link traversed by the packet. Each link (or hop) delay in turn consists of four components, including *processing delay, Queueing delay, transmission delay and propagation delay*. Fixed the packet length and the route, the packet round-trip delay only changes with the Queueing delay, which in Internet is changed with the fluctuation of traffic.

Internet is expanding dramatically fast, as it is the most complicated collection of Networks connected together. Many well developed and currently developing applications run across Internet, as mentioned in the previous section, to go around the global network. Some applications (audio, video) are sensitive to the performance of the whole Internet, or precisely, the Packet Delay in Internet.

When data networks become heavily utilized, the combined bandwidth demand from all sources occasionally exceeds the network capacity. For *data*, this is not a problem. It simply means that the data does not arrive as quickly; it is delayed by a few additional milliseconds. For data traffic, the motto is "Better late than never", so it does not have to pay the penalty of retransmission.

Video and voice traffic, on the other hand, must get a fairly regulated number of packets through to the destination in a timely manner. If video/voice packets are late enough to have missed their "play" window, they are useless. Hence the motto for video/voice traffic is "Better never than late. " The network should *drop* traffic that is late so it does *not consume additional scarce network resources*.

Quality of Service (QoS) techniques are designed to balance the needs of voice, video, and data across the network. QoS reserves a portion of the network bandwidth for the predictable use of the voice/video and lets the data traffic consume the remainder. By ensuring that bandwidth is available when needed for voice/video traffic, QoS techniques can reduce or eliminate audio pops, video artifacts, and other performance problems.

2.7 Internet Quality of Service

The phenomenal success of the Internet has created new *challenges*. Many new applications have very different requirements from those that the Internet was originally designed for. One challenge is *performance assurance*. The datagram model, on which the Internet is based, has few resource management capabilities inside the network and therefore cannot provide any resource guarantees to users, that is, one simply get what it gets. Another challenge is *service differentiation*. Because the Internet treats all packets the same way, it can only offer a *single* level of service. The applications, however, have diverse requirements. Interactive applications such as Internet telephony are sensitive to latency and packet losses. In contrast, a file transfer can tolerate a fair amount of delay and losses without much degradation of perceived performance. Customer requirements also vary depending on what the Internet is used for.

The capability to provide resource assurance and service differentiation in a network is often referred to as quality of service (QoS). Resource assurance is critical for many new Internet applications to flourish and prosper. The Internet will become a truly multiservice network only when service differentiation can be supported. Implementing these QoS capabilities in the Internet has been one of the toughest challenges in its evolution, touching on almost all

aspects of Internet technologies and requiring changes to the basic architecture of the Internet.[19]

The architectures and mechanisms developed in this regard address two key QoS issues in the Internet: *resource allocation* and *performance optimization*. *Integrated Services* and *Differentiated Services* are two resource allocation architectures for the Internet. The new service models proposed in them make possible resource assurances and service differentiation for traffic flows and users. *Multiprotocol Label Switching (MPLS)* and *traffic engineering*, on the other hand, give service providers a set of management tools for bandwidth provisioning and performance optimization; without which, it would be difficult to support QoS on a large scale and at reasonable cost.

2.7.1 Resource Allocation

Fundamentally, many problems seen in the Internet all come down to the issue of resource allocation. Packets get dropped or delayed because the resources in the network cannot meet all the traffic demands. A network, in its simplest form, consists of shared resources such as *bandwidth* and *buffers*, serving traffic from competing users. A network that supports QoS needs to take an active role in the resource allocation process and decides who should get the resources and how much.

The current Internet does not support any forms of active resource allocation. The network treats all individual packets exactly the same way and serves the packets on a first-come, first-serve (FCFS) basis. There is no admission control either-users can inject packets into the network as fast as possible.

The Internet currently *relies* on the *TCP protocol in the hosts* to detect *congestion* in the network and reduce the transmission rates accordingly. TCP uses a window-based scheme for congestion control. The window corresponds to the amount of data in transit between the sender and the receiver. If a TCP source detects a lost packet, it *slows* the transmission rate by reducing the window size by half and then increasing it gradually in case more bandwidth is available in the network. TCP-based resource allocation requires all applications to use the same congestion control scheme. Although such cooperation is achievable within a small group, in a network as large as the Internet, it can be easily abused.

The service that the current Internet provides is often referred to as *best effort*. Best-effort service represents the simplest type of service that a network can offer; it does not provide any form of resource assurance to traffic flows. When a link is congested, packets are simply pushed out as the queue overflows. Since the network treats all packets equally, any flows could get hit by the congestion. Although the best-effort service is adequate for some applications that can tolerate large delay variation and packet losses, such as file transfer and e-mail, it clearly does not satisfy the needs of many new applications and their users. New architectures for resource allocation that support resource assurance and different levels of services are essential for the Internet to evolve into a multiservice network. Over the last decade the Internet community came up with *Integrated Services* and *Differentiated Services*,

two new architectures for resource allocation in the Internet. These architectures introduced a number of new concepts and primitives that are important to QoS support in the Internet.

Integrated Services

The requirements of the real-time applications had major impacts on the architecture of Integrated Services. The Integrated Services architecture is based on *per-flow resource reservation*. To receive resource assurance, an application must make a *reservation* before it can transmit traffic onto the network. Resource reservation involves several steps. First, the application must characterize its traffic source and the resource requirements. The network then uses a routing protocol to find a path based on the requested resources. Next, reservation protocol is used to install the reservation state along that path. At each hop admission control checks whether sufficient resources are available to accept the new reservation. Once the reservation is established, the application can start to send traffic over the path for which it has exclusive use of the resources. Resource reservation is enforced by packet *classification* and *scheduling* mechanisms in the network elements, such as Routers.

Integrated Services architecture focused primarily on long-lasting and delay-sensitive applications. The World Wide Web, however, significantly changed the Internet landscape. Web-based applications now dominate the Internet, and much of Web traffic is short-lived transactions. Although per-flow reservation makes sense for long-lasting sessions, such as video conferencing, it is not appropriate for Web traffic. The overheads for setting up a reservation for each session are simply too high.

Differentiated Services

The Differentiated Services architecture was developed as an alternative resource allocation scheme for *service providers'* networks. The Internet community started to look for a simpler and more scalable approach to offer a *better than best-effort service*.

The solution is to develop a framework and standards for allocating *different levels of services* in the Internet. The new approach, called Differentiated Services, is significantly different from Integrated Services. Instead of making per-flow reservations, Differentiated Services architecture uses a combination of *edge policing*, *provisioning*, and *traffic prioritization* to make possible service differentiation.

In the Differentiated Services architecture, users' traffic is divided into a small number of forwarding *classes*. For each forwarding class, the amount of traffic that users can inject into the network is limited at the edge of the network. By changing the total amount of traffic allowed in the network, service providers can adjust the level of resource provisioning and hence control the degree of resource assurance to the users.

The edge of a Differentiated Services network is responsible for mapping packets to their appropriate forwarding classes. This packet classification is typically done based on the service level agreement (SLA) between the user and its service provider. The nodes at the edge of the network also perform traffic policing to protect the network from misbehaving traffic sources. Nonconforming traffic may be dropped, delayed, or marked with a different

forwarding class. The forwarding class is directly encoded into the packet header. After packets are marked with their forwarding classes at the edge of the network, the interior nodes of the network can use this information to *differentiate* the treatment of the packets. The forwarding classes may indicate drop priority or resource priority. For example, when a link is congested, the network will drop packets with the highest drop priority first.

Differentiated Services do *not* require resource reservation setup. The allocation of forwarding classes is typically specified as part of the SLA between the customer and its service provider, and the forwarding classes apply to traffic *aggregates* rather than to individual flows. These features work well with transaction-orientated Web applications. The Differentiated Services architecture also eliminates many of the scalability concerns with Integrated Services. The functions that interior nodes have to perform to support Differentiated Services are relatively simple. The complex process of classification is needed only at the edge of the network, where traffic rates are typically much lower.

2.7.2 Performance Optimization

Once the resource allocation architecture and service models are in place, the second issue in resource allocation is performance optimization; that is, how to organize the resources in a network in the most *efficient* way to maximize the probability of delivering the commitments and minimizes the cost of delivering the commitments. The connection between performance optimization and QoS support may seem less direct compared with resource allocation. Performance optimization is, however, an important building block in the *deployment* of QoS. Implementing QoS goes way beyond just adding mechanisms such as traffic policing, classification, and scheduling; fundamentally, it is about developing new services over the Internet.

The Internet's datagram *routing* was not designed for optimizing the performance of the network. *Scalability* and *maintaining connectivity* in the face of failures were the primary design objectives. Routing protocols typically select the shortest path to a destination based on some simple metrics, such as *hop count or delay*. Such simple approaches are clearly not adequate for supporting resource allocation. Simply using the shortest-path algorithm for selecting paths is likely to cause high rejection rate and poor utilization. The shortest-path routing does not always use the diverse connections available in the network. In fact, traffic is often unevenly distributed across the network, which can create congestion hot spots at some points while some other parts of the network may be very lightly loaded.

Performance optimization requires *additional capabilities in IP routing and performance management tools*. To manage the performance of a network, it is necessary to have explicit control over the paths that traffic flows traverse so that traffic flows can be arranged to maximize resource commitments and utilization of the network. MPLS *(Multi Protocol Label Switching)* has a mechanism called explicit routing that is ideal for this purpose. MPLS uses the label-switching approach to set up *virtual circuits* in IP-based networks. These virtual circuits can follow destination-based IP routing, but the explicit routing mechanism in MPLS also allows us to specify *hop by hop* the entire path of these virtual circuits. This provides a way to override the destination-based routing and set up traffic trunks based on traffic-engineering objectives.

The process of optimizing the performance of networks through efficient provisioning and better control of network flows is often referred to as *traffic engineering*. The basic problem addressed in traffic engineering is as follows: Given a network and traffic demands, how can traffic flows in the network be organized so that an optimization objective is achieved? The objective may be to *maximize the utilization* of resources in the network or to *minimize congestion* in the network. Typically the optimal operating point is reached when traffic is evenly distributed across the network. With balanced traffic distribution, both queuing delay and loss rates are at their lowest points. Obviously these objectives cannot be achieved through destination-based IP routing; there simply is not sufficient information available in IP routing to make possible such optimization.

Traffic engineering uses advanced route selection algorithms to provisioning traffic trunks inside backbones and arrange traffic flows in a way that maximizes the overall efficiency of the network. The common approach is to calculate traffic trunks based on flow distribution and then set up the traffic trunks as explicit routes with the MPLS protocol. The combination of MPLS and traffic engineering provides IP-based networks with a set of advanced tools for service providers to manage the performance of their networks and provide more services at less cost.

Chapter 3

Voice Over Internet Protocol (VoIP)

3.1 Introduction

Voice over IP, or VoIP, is simply the transfer of voice conversations as data over an IP network. The IP Network considered in this context of VoIP is *the Internet*. Internet uses packet switching for its communication. One fundamental characteristic of a packet-switched network is the *delay* encountered to deliver a packet from a source to a destination. Each packet generated by a source is routed to the destination via a sequence of intermediate nodes. The end-to-end delay is thus the sum of the delays experienced at each node or hop on the way to the destination. Each delay consists of two components, a *fixed* component, which includes the transmission delay at a node and the propagation delay on the link to the next node, and a *variable* component, which includes the processing and queuing delays at the node.

In real-time communication, the delay problem is also compounded by the need to remove *jitter*, which is a variable inter-packet timing caused by the network a packet traverses. Removing jitter requires collecting packets and holding them long enough to allow the *slowest* packets to arrive in time to be played in the correct sequence. This causes *additional delay*.

VoIP traffic must get a fairly regulated number of packets through to the destination in a timely manner. If these packets are late enough to have missed their "play" window, they are useless. Internet backbones and communication entities should have properly managed to stand up to the toll quality expectation of users.

3.2 How VoIP Works

One legacy of the Internet bubble is the idea that Internet Protocol (IP) can be used for more than just basic exchange of files and web content. IP offers enormous potential for communications. E-mail was the first (and still the most) successful communication application that leverages the Internet, and instant messaging has transformed the communication habits of many workgroups. Communication convergence is now being propelled by the increasing use of *Voice and Video* over data networks.

The new world movement in telephony is *convergence* and Voice over IP (VoIP). Convergence is simply uniting voice and data networks across a *single infrastructure*. A

converged network eliminates the need for duplicate hardware and special vendors who manage separate voice networks, often called PBXs (Private Branch Exchanges). VoIP uses the Internet protocol (IP) to deliver voice traffic (telephone calls) over a converged network. The major advantages of VoIP include efficient *Bandwidth utilization* and cost savings overtime through hardware reduction and eliminating the service changes and toll costs associated with old-world Telephony.

Voice communication over the Internet debuted in mid-1994 with a shareware program, 'Internet voice chat' for simple PC to PC connection. Today, this has evolved into an Internet Protocol (IP) telephony market which is capable of supporting the convergence of voice, video and data and becoming an ideal medium for new multimedia and advanced communication application ranging from Phone-to-Phone, PC-to-PC, PC-to-Phone, Phone-to-PC and Fax-to-Fax services- all over an *IP* network.

With a VoIP call, the call setup portion of the calling sequence has to be *simulated-dial tone, ringing, busy signals, etc*. The audio portion of the call itself needs to be converted from analog to digital, cut into packets, sent across the network still in packet format, reassembled, and converted from digital back to analog. Codecs (Coder/Decoders) at either end do the conversions from analog to digital and back.

To transfer voice and data on the Internet, the *same network* with e-mail and web traffic, a new and different set of components is required. Some of these components are:

- Codecs
- TCP/IP and VoIP protocols
- IP Telephony servers and PBXs
- IP Gateways and Routers
- IP phones and soft phones

Codecs

A codec (which stands for "compressor/decompressor"or coder/decoder") is the *hardware* or *software* that samples *analog* sound and converts it to *digital* bits, which it outputs at a predetermined data rate. The codec often performed *compression* to save bandwidth. There are dozens of available codecs, each with its own characteristics.

Table 6.1 lists some common VoIP codecs. The "*Nominal Data Rate*" column shows the rate at which the codec generates its output. The "Packetization delay" column refers to the delay a codec introduces as it converts from analog to digital and back.

Codec Name	***Nominal Data Rate***	***Packetization Delay***
G.711U	640kbps	1ms

G.711a	64.0 kbps	1ms
G.726-32	32.0 kbps	1ms
G.729	8.0 kbps	25 ms
G.723.1 MPMLQ	6.3 kbps	67.5 ms
G.723.1 ACELP	5.3 kbps	67.5 ms

Table 6.1. Common Codecs used in VoIP

The voice packet is constructed as a UDP/IP packet, to avoid TCP/IP's attempt to correct a corrupted packet by *retransmitting* the packet. Any attempt to retransmit a voice packet would introduce too much delay and be of no value. If FEC (forward error correction) is enabled, a corrupted or missing voice packet can be recreated from the FEC data stored in the previous voice packet. If FEC is not enabled, then corrupted packet is simply discarded, and the previous good packet is reused by the destination gateway.

But more information is needed on a packet-by-packet basis than UDP offered. So, for real-time or delay sensitive traffic, the Internet Engineering Task Force (IETF) adopted the RTP, Real time Transfer Protocol. VoIP rides on top of RTP, which rides on top of UDP. So VoIP is carried with an *RTP/UDP/IP* packet header.

VoIP Protocols

Application programs build their own families of "*higher layer" protocols* on top of the lower layer protocols they use for transport and other tasks. Implementing a VoIP telephone call on the data network involves the *call setup*-that is the VoIP equivalent of getting a dial tone, dialing a phone number, getting a ring or a busy signal at the far end, and picking up the phone to answer the call-and then the *telephone conversation* itself. VoIP protocols are required during both phases:

- Several higher layer protocols can accomplish *call setup and takedown*, including *H.323*, *SIP* (Session Initiation Protocol), *MGCP* (Media Gateway Control Protocol) and Magaco. The programs that implement the call setup protocols use TCP and UDP to encapsulate the data exchange during the call setup and takedown phases.

- The *exchange* of actual encoded voice data occurs after the call setup (and before the call takedown), using two data flows, one in each direction, to let both participants speak at the same time. Each of these two-way data flows uses a higher layer protocol called *Real-time Transport Protocol (RTP),* which is encapsulated in *UDP* as it travels over the wire. RTP, which is designed for applications that send data in one direction with *no acknowledgment,* is widely used for streaming audio and video. The header of each RTP datagram contains a time stamp, so the application receiving the datagram can reconstruct the timing of the original data. It also contains a sequence number, so the receiving side can deal with missing, duplicate, or out of order datagrams.

IP Telephony Servers and PBXs

Many data-networking transactions are based on the concept of *client/server* computing. Adding voice data to IP network provides yet another set of servers that are designed to provide voice service in new and innovative ways. An IP PBX (Private Branching eXchange) typically serves as the core IP Telephony server. On the PSTN, the PBX is often a "Closed-box" system-it provides all the voice functions and features you need- but usually in a *proprietary* manner. Management of the closed box platform is left up to the PBX vendor. With VoIP, an IP PBX can be built on a PC platform running on an Operating System like Microsoft Windows, Linux, or Sun Solaris. These platforms can be managed through vendor Application Programming Interfaces (APIs) and through the standard APIs provided by the Operating System itself. These IP PBXs provide the standard PSTN PBX service and more. A new concept introduced along with IP telephony servers is *clustering*, in which several of these servers are grouped together in a cluster to offer increased *scalability*, *reliability* and *redundancy*. Clustering is not available with traditional PBXs in the PSTN.

Gatekeepers are another type of servers. Gatekeepers are used by the *H.323* protocol to provide admission control features and other management functions for multimedia services.

VoIP gateways and Routers

VoIP Gateways and *IP Routers move* RTP voice datagrams through an IP network. VoIP Gateways provide a connection between the VoIP networks and the PSTN. These devices therefore play a key role in the migration path towards VoIP. There are few totally VoIP phone networks in the world today. It is necessary to connect to the PSTN to place calls to PSTN users. VoIP gateway uses the PSTN to place calls to PSTN users. VoIP Gateways may also provide conversion between different codecs, which is called "Transcoding".

IP Phones and Soft Phones

To make VoIP work, *analog* audio must first be converted to digital datagrams. This process is done by codecs. If you are still using older analog telephones, the codecs are located in the IP PBX. Incoming calls are digitalized there, before being forwarded to the IP network.

In new digital telephones, *IP Phones,* the codecs can be located in the telephones themselves. An IP phone makes the data connections to an IP telephony server, which does the call setup processing. Also, S*oft Phones, computers* with the corresponding software can serve the role of an IP phone. As with an IP phone, your computer probably relies on an IP telephony server to do call set up processing. Fig 6.1 shows the audio data flow in the Internet.

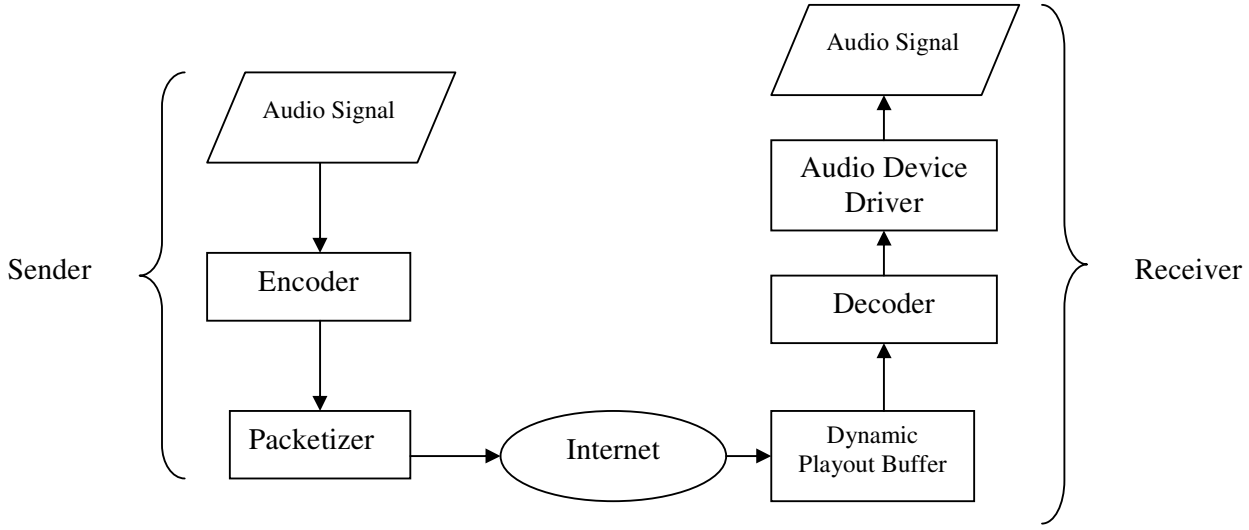


Fig. 3.1 Audio Data Flow over the Internet

To pave the way for VoIP finally replace PSTN, any implementation of VoIP should resolve major concerns related to VoIP. This can be expressed in two ways: user satisfaction must get guaranteed by keeping the voice quality at an acceptable range and security of data flow must kept at high level while running on a public network.

IP based networks are developed for *data* applications and do not provide true real-time capabilities for services such as *voice* and *video*. When IP networks get heavily loaded the voice packets queue up, creates delayed packets that degrade the voice quality. There is also an end-to-end delay usually comes from the voice Gateway because of *compression*, *decompression*, *packetization* and *depacketization*. To counteract the degradation in voice quality, therefore, proper Q*uality of Service* techniques should be implemented.

In a public Internet the packets can traverse through any router and can be intercepted by any one. This, therefore, requires implementation of proper security mechanisms. The security solutions must involve in *four* areas: User and data authentication, Data privacy, Access control and Policy Management.

3.3 Packet Delays and VoIP

Although a VoIP call consumes relatively *little bandwidth* and uses small packet sizes, voice applications have strict requirements for *delay, jitter and lost data*, while other application require high *throughput* or low response times. Voice traffic must get a fairly *regulated* number of packets through to the destination in a timely manner. If packets must not be late to have missed their “play” window, as they are useless otherwise.

Both the Packet Round Trip delay and the one-way end-to-end packet delay have their significance on VoIP traffic. A longer packet Round Trip Delay causes a phenomenon called *Echo*, and a longer one-way end-to-end packet delay cause *talker over lap* and *packet loss*. . Echo becomes a significant problem when the round trip delay becomes greater then *50 milliseconds* [11]. *Talker overlap* (or the problem of one talker stepping on the other talker’s speech) will have a significant impact on voice quality when the one-way delay exceeds *250ms*. And *Packet Loss* has occurred when an end-to-end delay of a packet becomes prolonged so that a packet lost its "play window". The network should *drop* traffic that is late so it does not consume additional network resources.

The end-to-end delay budget is therefore the major constraint and driving requirement for reducing delay through a packet network. The International Telecommunications Union Telecommunication standardization sector (ITU-T) *G.114* recommendation specifies that for good voice quality, no more than 150 ms of one-way, end-to-end delay should occur [11].

As described in the previous section the end-to-end delay consists of two components, *fixed* and *variable;* but Quality of Service (QoS) provisioning is used only to improve the variable delays.

Accumulated Delay is caused by the need to collect a frame of *voice samples* to be processed by the voice coder. It is related to the type of voice coder used and varies from a single sample time (0.125ms) to many milliseconds. *Processing Delay* is a delay caused by the actual process of encoding and collecting the encoded samples into a packet for transmission over the packet network. The encoding delay is a function of both the *processor execution time* and the *type of algorithm used.* Often, multiple voice coder frames will be collected in a single packet to reduce the packet network overhead. *Propagation Delay* is the delay caused due to signal *propagation* in the medium used. It is variable based on the capacity of the *medium of propagation*. *Queueing Delay* is resulted when packets are held in queue because of congestion on an outbound interface, which occurs when more packets are sent out than the interface can handle at a given interval. In an unmanaged, congested network, queuing delay can add up to two seconds of delay (or result in the packet being dropped). This lengthy period of delay is unacceptable in almost any voice network. *Jitter Delay* is created by the *need to remove jitter*, which is the variation of packet inter arrival time. Jitter is one issue that exists only in packet-based networks. Removing jitter requires collecting packets and *holding* them long enough to allow the slowest packets to arrive in time to be played in the correct sequence, which causes jitter delay.

Every VoIP network should *minimize* variable delays for the *voice packets* to achieve the toll voice quality of the PSTN systems, and consequently made possible the migration from PSTN to VoIP. The quality improvement is achieved through usage of different Quality of Service (QoS) provisioning techniques. Section 3.4 discusses some of the *existing* QoS techniques implemented in VoIP.

3.4 Quality of Service for VoIP

This section *specifically* iterates ways for improving *voice quality* of VoIP networks through the use of Quality of Service (QoS) techniques. The Internet QoS techniques discussed in Chapter 2 (section 2.7) are *generalizations* of the QoS techniques to be discussed in this section.

VoIP comes with its own set of problems-namely *delay, jitter and packet loss* as discussed in the previous section. QoS can help to relieve these problems. Some of the problems QoS *cannot* solve are propagation delay, codec delay, sampling delay, and digitization delay.

3.4.1 Queueing

To minimize degradation in voice quality, delays at intermediate queues of the voice packets should be *managed*. Also the packet rejection probability at the queues has to be kept very small. The following subsections discuss some ways of managing queues for better performance.

Weighted Fair-Queueing (WFQ)

First in first out (*FIFO*) Queueing places all packets it receives in *one Queue* and transmits them as bandwidth becomes available. WFQ, on the other hand, uses *multiple Queues* to *separate* flows and gives equal amount of bandwidth to each flow. This prevents one application from consuming all available bandwidth.

WFQ ensures that queues do not starve for bandwidth and that traffic gets predictable service. Low volume data streams receive preferential service, transmitting their entire offered loads in a timely fashion. High volume traffic streams share the remaining capacity, obtaining of equal or proportional bandwidth.

Fair Queueing *dynamically* identifies data streams or flows based on several factors. These data streams are *prioritized* based up on the amount of bandwidth that the flow consumes. This algorithm enables bandwidth to be shared fairly, with out the use of access lists or other time consuming administrative tasks. It also enables low-band width applications, which make up most of the traffic, to have as much bandwidth as needed, relegating higher band

width traffic to share the remaining traffic in a fair manner. Fair Queueing offers *reduced delay and jitter* and enables efficient sharing of available bandwidth between all applications.

Priority Queueing (PQ)

PQ enables configuration of *four* traffic *priorities*; *high*, *normal*, *medium,* and *low.* Inbound traffic is assigned to one of the four output Queues. Traffic in the high-priority queue is serviced until the queue is empty, then packets in the next priority queue are transmitted.

This Queueing arrangement ensures that *mission critical* traffic is always given as much bandwidth as it needs; however, it *starves* other applications to do so. PQ is best used when the highest priority traffic consumes the *least* amount of link bandwidth.

Class Based Weighted Fair Queueing (CB-WFQ)

CB-WFQ has all the benefits of WFQ, with the additional functionality of providing granular support for defined *classes* of Traffic. CB-WFQ enables us to define what constitutes a class based on criteria that exceeds the confines of flow. Using CB-WFQ, one can create specific class for voice traffic, through the use of access lists. These classes of traffic determined how packets are grouped in different queues.

The other feature of CB-WFQ is that it enables the network administrator to specify the *exact* amount of bandwidth to be allocated per class of traffic. With CB-WFQ, each class is associated with a *separate* Queue. One can allocate a specific minimum amount of guaranteed bandwidth to the class as a percentage of the link. Other classes can share unused bandwidth in proportion to their assigned weights. Classification tools mark a packet or flow with a specific priority. This making establishes a *trust* boundary that must be enforced.

Low Latency Queueing (LLQ)

Queueing is one of the most important mechanisms for ensuring *voice quality* within a data network. This is more vital on small bandwidth links, such as WAN links, because multiple traffic flows are contending for a very limited amount of network bandwidth. The contention can be relieved by implementing proper traffic classification mechanisms. Once traffic has been classified, the flow can be placed into an interface Queue that meets its handling requirements. VoIP should be placed on a *Priority Queue (PQ).* However, *other traffic* types may have also specific bandwidth and delay characterstics as well. These requirements might be addressed with *LLQ.* LLQ *combines* the use of a PQ with CB-WFQ. Classes are defined with classification admission schemes. *Traffic Flows* will have access to either the PQ, one of the class-based Queues, or a default Weighted Fair Queue.

For low-speed WAN connections, it is also necessary to provide a mechanism for *Link Fragmentation and Interleaving (LFI).* A data from the Queue can be sent to the physical wire only at the *Serialization Rate* of the interface. The Serialization Rate is the size of the frame divided by the clocking speed of the interface. If a delay-sensitive voice packet is behind a large data packet in the interface Queue, the end-to-end delay budget of *150ms* could be

exceeded. In addition, even relatively small frames can adversely affect overall voice quality by simply *increasing* the jitter to a value greater than the size of the adaptive *jitter buffer* at the receiver. LFI tools are used to *Fragment* large data frames into regularly sized packets and to *Interleave* voice frames in to the flow so that the end-to-end delay predicted accurately. Figures 3.2 and 3.3 show the use of LLQ and LFI.

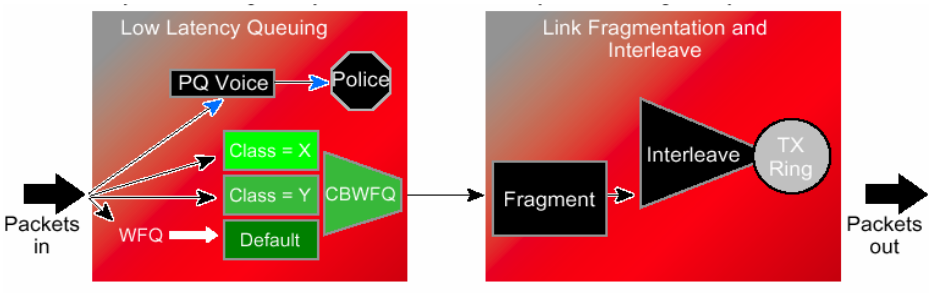


Fig. 3.2 Queueing with LLQ and LFI (Source: Reference [18])

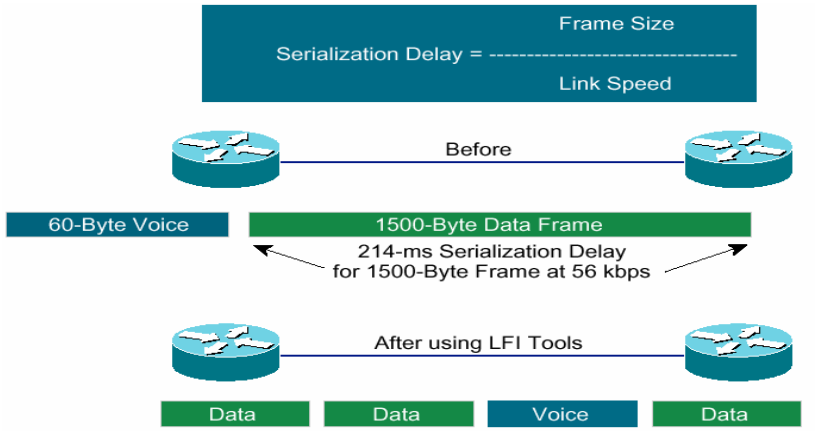


Fig.3.3:Using LFI tools to reduce packet delay due to larger frames. (Source: Reference[18])

3.4.2 Jitter

Jitter is a *variable inter-packet timing* caused by the network a packet traverses. Removing jitter requires collecting packets and *holding* them long enough to allow the slowest packets to arrive in time to be played in the correct sequence. This causes additional delay. An important component at the receiving end is the *jitter or playback* buffer whose purpose is to absorb *variations* in delay and provide a smooth playout. This is achieved by *holding* arriving packets until a later playout time in order to ensure that there are enough packets buffered to be played out continuously. Any packet arriving after its scheduled playout time is discarded. Clearly, there exists a trade-off between delay and loss. The playback buffer may operate in one of two modes: *fixed* or *adaptive*.

A fixed scheme schedules the playout of a packet after a fixed (network and buffering) delay from its sending time, the *same* for all packets. The value of this fixed delay is important in order to avoid either unnecessarily delaying or dropping of packets. It should be chosen based

on some knowledge of the delay on the path. However, such an assessment may not always be possible or the statistics of the network delay itself may change with time. In addition, a fixed playback scheme needs *synchronization* between the source and the receiver in order to guarantee the chosen end-to-end fixed delay.

The two *conflicting* goals of minimizing delay and removing jitter have engendered various schemes to adapt the *jitter buffer size* to match the time varying requirements of network jitter removal. This adaptation has the explicit goal of *minimizing* the size and delay of the jitter buffer, while at the same time preventing buffer underflow caused by jitter. One approach for *adapting* the playback buffer is to *measure the variation* of packet level in the jitter buffer over a period of time, and *incrementally* adapt the buffer size to match the calculated jitter. The other approach is to *count* the number of packets that arrive *late* and create a ratio of these packets to the number of packets that are successfully processed. This ratio is then used to *adjust* the jitter buffer to target a predetermined allowable late packet ratio. This approach works best with the networks with highly variable inter arrival intervals, such as Internet.

3.4.3 Echo

Echo is the reflection of the participants' signals, perceived as delayed and attenuated versions of their own voices. The *larger* the end-to-end delay, the more annoying is the echo. Although one might at first think that echo cannot happen in a packetized voice system, reflections may indeed happen (i) at the four-to-two wires hybrid connection between a packet and a circuit switched network and (ii) at the PC end-point when the microphone picks up the remote person's voice from the speaker as well as multiple reflections in the room and bounces them back. Both types of echo can be controlled by an *Echo Canceller* that should be located as close to the source of echo as possible.

Chapter 4

Self-Similar and Long-Memory Processes

4.1 Introduction

Since the seminal study of Leland, et al [1], which set a ground work for considering self-similarity an important notion in understanding of network traffic including the modelling and analysis of Network performance, an explosion of work has ensued investigating the multifaceted nature of this phenomenon. The long held paradigm in the communication and performance communities has been that voice traffic, and by extension, data traffic is adequately described by certain Markovian, Models (e.g. Poisson), which are amenable to accurate analysis and efficient control.

The first reason stems from the well developed field of Markovian Analysis, which allows tight equilibrium bound on *performance variables* such as the waiting times in various queuing systems to be found. This also forms a pillar of performance analysis from the Queueing theory side. The second feature is, in part, due to the simple correlation structure generated by Markovian sources whose performance impact-for example, as affected by the likelihood of prolonged occurrence of "bad events" such as concentrated packet arrivals-is fundamentally well behaved. Specifically, if such processes are appropriately *rescaled* in time, the resulting coarsified processes rapidly lose dependence, taking on the property of an independent and identically distributed (i.i.d.) sequence of random variables with its associated niceties. Principal among them is the exponential smallness of rare events, a key observation at the center of large deviations theory.

The behaviour of a process under rescaling is an important consideration in performance analysis and control since buffering and, to some extent, bandwidth provisioning can be viewed as operating on the rescaled process. The fact that Markovian systems admit to this avenue of taming variability has helped shape the optimism permeating the late 1980s' and early 1990s regarding the feasibility of achieving efficient traffic control for quality of service (QoS) provisioning. The discovery and, more importantly, succinct formulation and recognition that data traffic may not exhibit the here to accustomed scaling properties has significantly influencing, necessitating a reexamination of its fundamental premises. [2]

4.2 What is Self Similarity?

Self-Similarity and Fractals, notions pioneered by Beniot Mandelbrot,[3] describe the phenomenon where a certain property of an object-for example, a natural image, the convergent sub domain of certain dynamical system, a time series (the mathematical object of our interest)-is preserved with respect to scaling in space and /or time. If an object is self-similar or fractal, its parts, when magnified, resemble, - in a suitable sense - the shape of the whole.

For example, the two dimensional (2D) cantor set living on A= $[0,1]\ X[0,1]$ is obtained by starting with a solid or black unit square, scaling its size by 1/3, then placing four copies of the scaled solid square at the four corners of A. If the same process of scaling followed by translation is applied recursively to the resulting objects at infinitum, the limit set thus reached defines the 2D cantor set. This constructive process is illustrated in figure 4.1. The limiting object-defined as the infinite intersection of the iterates-has the property that if any of its corners are "blown up" suitably, then the shape of the zoomed-in part is similar to the shape of the whole, that is, it is self-similar. Of course, this is not too surprising since the constructive process - by its recursive action –endows the limiting object with the scale invariance property.

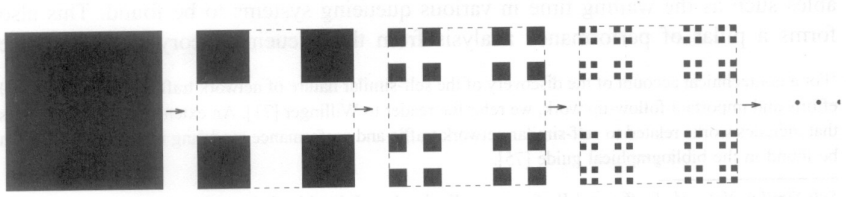


Fig 4.1 Two-Dimensional Cantor Set

The one dimensional (1D) cantor set, for example, as obtained by projecting the 2D cantor set onto the line, can be given an interpretation as a traffic series $X(t) \in [0,1,]$ -call it "Cantor Traffic"-*where* $X(t) = 1$ means that there is a packet transmission at time t. This is depicted in Fig 4.2 (a). If the constructive process is terminated at iteration $n \geq 0$,then the contiguous line segments of *length* $1/3^n$ may be interpreted as *On Periods* or packet trains of *duration* $1/3^n$ and the segments between successive on periods as O*ff Periods* or absence of traffic activity. Non-Uniform traffic intensities may be imparted by generalizing the constructive framework via the use of probability measures. For example, for the 1D cantor set, instead of letting the left and right components after scaling have identical "mass", they may be assigned different measures, subject to the constraint that the total mass be preserved at each stage of the iterative construction. This modification corresponds to defining a probability measure μ on the Borel subsets of $[0,1]$ and distributing the measure at each iteration non-uniformly left and right. Cantor set construction -viewed as a map-is not measure preserving. Figure 4.2 (b) shows such a construction with weights $\alpha_L = 2/3, \alpha_R = 1/3$ for the left and right components, respectively. The probability measure is represented by 'height'; we observe that scale invariance is preserved. In general the traffic patterns producible with $\alpha L, \alpha R$ are limited, but one can extend the framework by allowing possibly different weights associated with every edge in the weighted binary tree induced by the 1D cantor set construction.

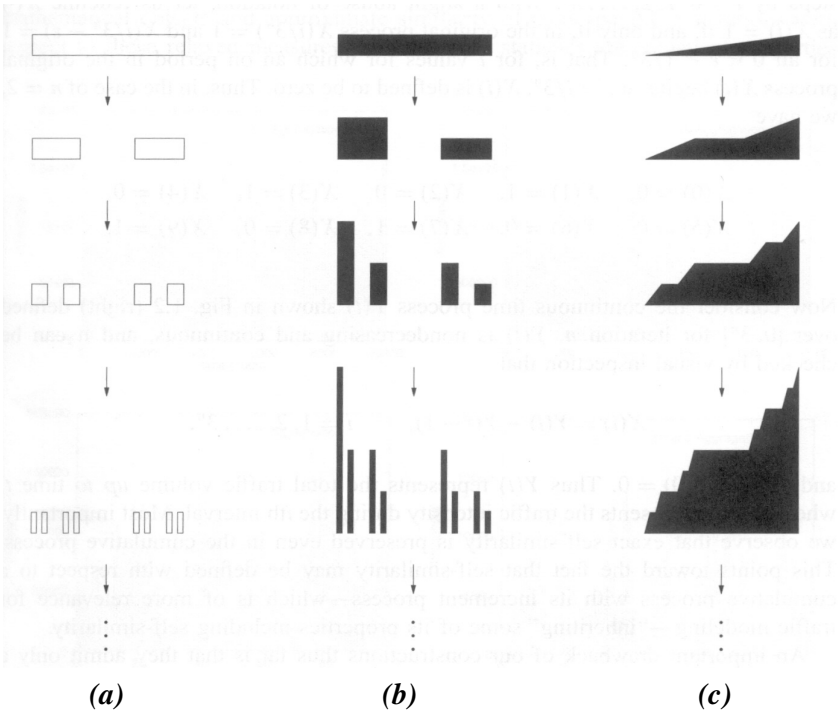


Fig. 4.2 (a) 1D Cantor set interpreted as On/Off Traffic. (b)Non-uniform

Where as the previous constructions are given interpretations as traffic activity *per unit time*, we will find it useful to consider their corresponding *cumulative* processes, which are non-decreasing processes whose differences-also called Incremental processes -constitute the original process. For example, for the On/Off cantor traffic construction (refer Fig 4.2.), let us assign the interpretation that time is discrete such that at step $n \geq 0$, it ranges over the values t=0, $1/3^n, 2/3^n, \ldots\ldots, (3^n-1)/3^n, 1$. Thus we can equivalently index the discrete time steps by i=0,1,2,…., 3^n. With a slight abuse of notation, let us redefine X(•) as X (i)=1 *if, and only if*, in the original process $X\left(\frac{i}{3^n}\right)=1$ and $X(\frac{i}{3^n}-\varepsilon)=1$ for all $0<\in<\frac{1}{3^n}$. That is, for *i* values for which an ON period in the original process X(t) begins at $t=i/3^n$, $X(i)$ is defined to be zero. Thus, in the case of n=2, we have

$$X(0)=0, X(1)=1, X(2)=0, X(3)=1, X(4)=0$$
$$X(5)=0, X(6)=1, X(7)=0, X(8)=1, X(9)=0$$

Now consider the continuous time process **Y(t)** shown in Fig 4.2.(c) defined over $[0,3^n]$ for iteration *n*. Y(t) is non-decreasing and continuous, and it can be cheeked by visual inspection that: *X(i)=Y(i)-Y(i-1),i=1,2,……,* 3^n, and *X(0)=Y(0)=0*. Thus Y(t) represents the *total* traffic volume up to time t, where as *X(i)* represents the traffic intensity during the i^{th} interval. Most importantly, we observe that the exact self-similarity is preserved even in the cumulative process. This points toward the fact that self-similarity may be defined with respect to a cumulative process with its increment process-which is of more relevance for traffic modeling-"inheriting" some of its properties including self-similarity.

An important draw back of the construction thus far is that they admit only a strong form of recursive regularity-that of *deterministic* self-similarity-and needs to be further generalized for traffic modeling purposes where *stochastic* variability is an essential component.

4.3 Stochastic Self-Similarity

Stochastic Self-Similarity admits the infusion of non-determinism as necessitated by measured traffic traces but, nonetheless, is a property that can be illustrated visually. Fig 4.3 (a) shows a traffic trace, where throughput in Bytes plotted against time where time granularity is 100s. That is, a single data point is the aggregated traffic volume over a 100 second interval, Figure 4.3. (b) is the same traffic series whose first 1000 second interval is ”blown up” by a factor of ten. Thus, the truncated time series has a time granularity of 10s. The remaining two plots (Fig 4.3. (c) and (d)) zoom in further in the initial segment by rescaling successively by factor of 10, for 1s and 10ms granularity respectively.

Unlike deterministic fractals, the objects corresponding to Fig 4.3 do not posses resemblance of their parts with the whole at finer details. Here, we assume that the measure of “resemblance” is the shape of the graph with the magnitude suitably normalized. Indeed, for measured traffic traces, it would be too much to expect to observe exact, deterministic self-similarity given the stochastic nature of many network events (e.g. Source arrival behavior) that collectively influence actual network traffic.

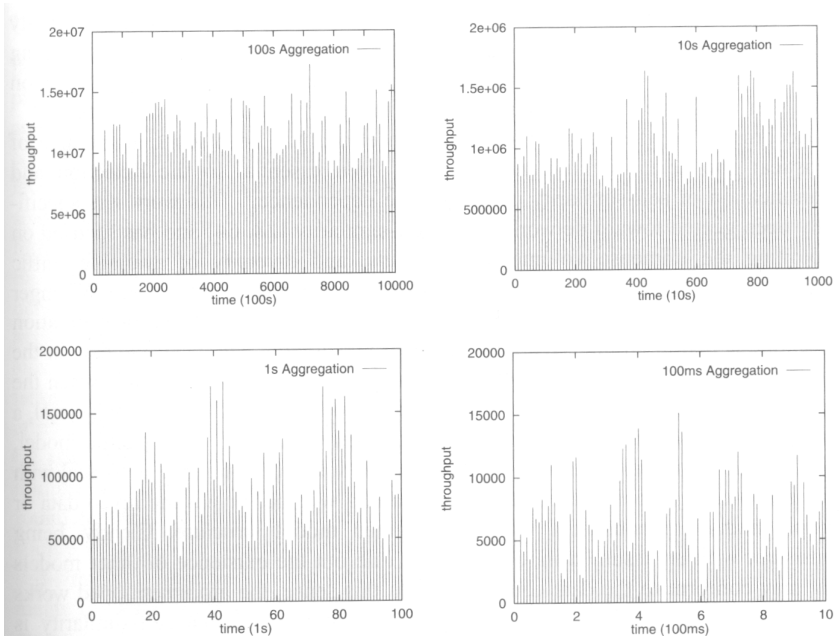


Fig. 4.3 Stochastic Self-Similarity-in the "Burstiness preservation

If we adopt the view that traffic series are sample paths of stochastic processes and relax the measure of resemblance, say, by focusing on certain *statistics* of the rescaled time series, then

it may be possible to expect exact similarity of the *mathematical objects* and approximate similarity of their specific realizations with respect to these relaxed measures.

Second order statistics are statistical properties that capture *burstiness* or *variability*, and the *autocorrelation* function is a yardstick with respect to which scale invariance can be fruitfully defined. The shape of the autocorrelation function-above and beyond its preservation across rescaled time series-will play an important role. In particular, correlation, as a function of time lag, is assumed to decrease polynomially as opposed to exponentially. The existence of non-trivial correlation "*at a distance*" is referred to as *Long Range Dependence.*

4.4 Technical Background

4.4.1 Second order self-similarity and Stationarity

Consider a stochastic process or time series X(t), $t \in Z$, where X(t) can interpreted as the traffic volume - measured in *packets*, *bytes* or *bits* - at time instance t, say from time 0.

To minimize confusion, when a "Cumulative" view is taken, we will denote the process by Y(t). Then reserve X(t) to be the increment process corresponding to Y(t), that is *X(t)=Y(t)-Y(t-1)*.

For traffic modeling purposes, we would like *X*(t) to be "Stationary" in the sense that its behaviour or structure is invariant with respect to shifts in time. In other words, t's responsibility as an absolute reference frame is relieved. Without some form of Stationarity, "anything" is allowed and a model uses much of its usefulness as a compact description of (assumed) tractable phenomenon. X(t) is strictly stationary if $(X(t_1), X(t_2), \ldots\ldots, X(t_n))$ *and* $(X(t_{1+k}), X(t_{2+k}), \ldots\ldots, X(t_n + k))$ posses the same joint distribution for all $n \in Z_+$ *and* $t_1, t_{2,} \ldots\ldots, t_n, k \in Z$.

Denoting the k-shifted process or time series as X_k; X and X_k are said to be *equivalent* in the sense of finite dimensional *distribution*, can be expressed as $X =_d X_k$.

Imposing strict Stationarity is too restrictive and we will be interested in a *weaker* form of Stationarity, called *Second Order Stationarity*[1], which requires that the *Autocovariance* function $\gamma(r,s) = E\{X(r) - \mu)(X(s) - \mu)\}$ satisfies translation invariance, that is, $\gamma(r,s) = \gamma(r+k, s+k)$ for all *r, s,* $k \in Z$.

The first two moments are assumed to exist and be finite and set $\mu = \mathrm{E}[X(t)]$ and $\sigma^2 = \mathrm{E}[(X(t) - \mu)^2]$ for all $t \in Z$. It is also assumed $\mu = 0$. Since by Stationarity, $\gamma(r,s) = \gamma(r-s, 0)$, we denote the Autocovariance by $\gamma(k)$.

[1] Equivalent names are Weak, Covariance and Wide sense stationary.

To formulate *scale invariance*, first define the *aggregated* process $X^{(m)}$ of X at aggregation level *m* as

$$X^{(m)}(i) = \frac{1}{m} \sum_{t=m(i-1)+1}^{mi} X(t) \qquad (4.1)$$

That is, *X(t)* is partitioned into non-overlapping blocks of size ***m***, their values are averaged and *i* is used to index those blocks.

Let $\gamma^{(m)}(k)$ denote the *autocovariance* function of $X^{(m)}$, under the assumption of Second order Stationarity one can arrive at the following definition of Second Order Self-Similarity.

Definition 4.1: (*Second Order Self-Similarity) X(t)* is *exactly* Second Order Self-Similar with Hurst Parameter $H, (1/2 < H < 1)$ if

$$\gamma(k) = \sigma^2 \big/ 2 \left((k+1)^{2H} - 2k^{2H} + (k-1)^{2H} \right) \qquad (4.2)$$

for all $k \geq 1$.

X(t) is *asymptotically* Second Order Self-Similar if

$$\lim_{m \to \infty} \gamma^{(m)}(\mathrm{K}) = \sigma^2 \big/ 2 \left((k+1)^{2H} - 2k^{2H} + (k-1)^{2H} \right) \qquad (4.3)$$

It can be checked that Eq(4.2) implies $\gamma(k) = \gamma^{(m)}(k)$ for all $m \geq 1$.Thus second order self similarity captures the property that the correlation structure is exactly (Eq (4.2)) or asymptotically (the weaker Eq (4.3)) preserved under time aggregation.

The form of $\gamma(k) = \sigma^2 \big/ 2 \left((k+1)^{2H} - 2k^{2H} + (k-1)^{2H} \right)$ is not accidental and implies further structure, called *Long Range Dependence,* which is defined later. Second Order Self-Similarity (in the exact or asymptotic sense) has been a dominant framework for *Modeling* Network Traffic.

4.4.2 Distributional Self-Similarity

To understand the particular form of $\gamma(k)$ in the definition of Second Order Self-Similarity, I will make a short detour and discuss self-similar processes in slightly more generality.

Consider the Cumulative process *Y(t)*, albeit in continuous time $t \in R$. Following is a definition of self-similarity for *continuous-time* processes in the sense of finite dimensional distributions.

Definition 4.2: *Y(t)* is Self-Similar with Self-Similarity parameter, that is , *Hurst* parameter, $H, (0 < H < 1)$, denoted ***H-SS***, if for all *a>0* and $t \geq 0$

$$Y(t) =_d a^{-H} Y(at) \qquad (4.4)$$

where $=_d$ *imply equivalence in distribution sense.*

Thus *Y(t)* and its time scaled version, after normalized by a^{-H}, must follow the same distribution.

In the traffic modeling context, it is convenient to think of *Y(t)* as the cumulative or total traffic up to time t. For $a > 1$, time is stretched or dilated Therefore, *a contraction factor* a^{-H} is applied to make the magnitude of $Y(at)$ comparable to that of *Y(t)*. For $a < 1$, the opposite holds true.

As a varies, the scaling exponent *H* remains invariant. This is a most natural definition; however, it has an important drawback. Unless it is *degenerate*, that is *Y(t)=0* for all $t \in R$, *Y(t)* can not be stationary due to the normalization factor a^{-H}. Its increment process $X(t) = Y(t) - Y(t-1)$, however, is another matter. In particular, consider the case where *Y(t)* is H-SS and has *Stationary increments*; in this case we say *Y(t)* is H-SSSi.

Let us further assume that *Y(t)* has finite variance. It can be checked that $E[Y(t)] = 0, E[Y^2(t)] = \sigma^2 |t|^{2H}$, and

$$\gamma(k) = \frac{\sigma^2}{2}\left(|t|^{2H} - |t-s|^{2H} + |s|^{2H}\right) \qquad (4.5)$$

This is achieved by noting that[2]

$$Y(t) =_d t^H Y(1) \qquad (4.6)$$

from which it follows that

$$E[Y^2(t)] = \sigma^2 t^{2H} \qquad (4.7)$$

Eq(4.7), then, can be used in the derivation of the auto covariance function of Eq(4.5) The increment process *X(t)* has *mean 0* and *Autocovariance* $\gamma(k)$ as given in equation (4.2).

[2] From $a^H Y(t) =_d Y(at)$, substitute t=1 and $a = t$

How does distributional self-similarity (of a *continuous time process*) tie in with Second-Order Self-Similarity (of a *discrete time process*), which requires exact or asymptotic invariance with respect to second order statistical structure of the aggregated time series $X^{(m)}$? A key observation lies in noting that $X^{(m)}$ can be viewed as computing a *Sample Mean*.

$$X^{(m)} = \frac{1}{m}\sum_{t=1}^{m} X(t) = m^{-1}(Y(m) - Y(0)) =_d m^{-1} m^{H}(Y(1) - Y(0)) =_d m^{H-1} X \qquad (4.8a)$$

Thus, if *Y(t)* is an H-SSSi process, then its increment process *X(t)* satisfies.

$$X =_d m^{1-H} X^{(m)} \qquad (4.8b)$$

which shows how $X^{(m)}$ is relate to *X* via a simple scaling relationship involving *H* in the sense of finite dimensional distribution.

Equations (4.2) and (4.3), then, express the fact that X and $m^{1-H} X^{(m)}$ are required to have exactly or asymptotically the *same second order structure*. As a result, depending on whether a discrete time process *X(t)* satisfies Eq (4.8b) for all $m \geq 0$ or only in the limit as $m \to \infty$, *X(t)* is said to be Exactly Self-Similar or Asymptotically Self-Similar.

As a lead-in to the *role* of the parameter *H*, recall that the variance of the sample mean $\overline{Z}$ of a random variable Z satisfies $Var(\overline{Z}) = \sigma^2 / m$, where *m* is the sample size.

From Eq(4.8b) it follows that

$$Var\left(X^{(m)}\right) = \sigma^2 m^{2H-2} \qquad (4.9)$$

When viewed as a sample mean, where the samples are drawn *independently*, $Var(X^{(m)})$ reduces to $\sigma^2 m^{-1}$ if $H = 1/2$.

If $H \neq 1/2$, in particular, $1/2 < H < 1$, then:

$$Var(X^{(m)}) = \sigma^2 m^{-\beta} \qquad (4.10)$$

with $0 < \beta < 1$ (and $H = 1 - \beta/2$), hints at a certain *dependency* structure in the **"***Samples***"** (i.e. time series in our case) that causes $Var(X^{(m)})$ to converge to zero slower than the rate m^{-1}.

4.4.3 Long Range Dependence.

Thus far the focus is on explicating the role of Self-Similarity in the second-order stationary and distributional senses with little regard to role of *H* and its range of values. Let us return to the definition of Second-Order Self-Similarity and its Autocovariance $\gamma(k)$.

Let $r(k) = \gamma(k)/\sigma^2$ denote the *Autocorrelation* function. For $0 < H < 1, H \neq 0.5$ it holds

$$r(k) \sim H(2H-1)k^{2H-2},\ k \to \infty \qquad (4.11)$$

In particular, if $1/2 < H < 1$, $r(k)$ asymptotically behaves as $Ck^{-\beta}$, for $0 < \beta < 1$, where *C>0* is a constant, and $\beta = 2 - 2H$. Then it implies:

$$\sum_{k=-\infty}^{\infty} r(k) = \infty \qquad (4.12)$$

That is, the autocorrelation function decays slowly (that is, *Hyperbolically*), which is the essential property that causes it to be not summable. When $r(k)$ decays hyperbolically, we call the corresponding stationary process *X(t) Long range Dependent*. *X(t)* is *Short Range Dependent* if the Autocorrelation function is summable.

An essential equivalent definition can be given in the frequency domain where the spectral density.

$$\Gamma(\nu) = (2\pi)^{-1} \sum_{k=-\infty}^{\infty} r(k) e^{jk\nu} \qquad (4.13a)$$

is required to satisfy the property:

$$\Gamma(\nu) \sim c|\nu|^{-\alpha},\ \nu \to 0 \qquad (4.13b)$$

Here *c>0* is a constant and $0 < \alpha = 2H - 1 < 1$. Thus $\Gamma(\nu)$ diverges around origin, implying ever larger contributions by low frequency components.

Following are some simple facts regarding the value of H and its impact on $r(k)$. First, if $H = 1/2$, then $r(k) = 0$, and *X(t)* is trivially Short-Range Dependent by virtue of being completely uncorrelated. In the case where $0 < H < 1/2$ we have $\sum_{k=-\infty}^{\infty} r(k) = 0$, an artificial condition rarely encountered in applications. *H=1* is uninteresting since it leads to the

degenerate situation $r(k)=1$ for all $k \geq 1,$. Finally, H-values bigger than 1 are prohibited due to the stationary condition on *X(t).*

4.4.4 Self-Similarity Versus Long Range Dependence

The proceeding discussions indicate that there are Self-Similar processes that are not Long-range Dependent, and vice versa. For example, Brownian motion is $1/2-SSSi$ with white Gaussian noise as its *increment* process, but the latter is not long range dependent. Conversely, certain fractional ARIMA time series generate long-range dependence but they are not self similar in the distribution sense.

In the case of asymptotic second order self-similarity however, by the restriction $0.5 < H < 1$ in the definition, Self-Similarity implies Long Range Dependence, and vise versa. It is for this reason and the fact that asymptotic second-order Self-Similar Processes are employed as "Canonical" traffic models, that I use Self-Similarity and Long-Range dependence *interchangeably* when the context does not lead to confusion.

4.5 Impact of Heavy Tails

4.5.1 Heavy Tailed Distribution

There is an intimate relationship between heavy-tailed distributions and Long-Range Dependence.

A *random variable Z* has a *Heavy-Tailed Distribution* if

$$\Pr[z > x] \sim cx^{-\alpha}, x \to \infty \qquad (4.14)$$

Where $0 < \alpha < 2$ is called the *Tail Index* or *Shape Parameter* and *c* is a positive constant.

This is in contrast to Light-Tailed distributions-for example *Exponential and Gaussian*-which possess an exponentially decreasing tail.

A distinguishing mark of heavy tailed distributions is that they have infinite variance for $0 < \alpha < 2,$ and if $0 < \alpha \leq 1,$ they also have an unbounded mean. In the Networking context, we will be primarily interested in the case $1 < \alpha < 2.$

A frequently used heavy-tailed distribution is the *Pareto* distribution whose distribution function is given by

$$\Pr[z \leq x] = 1 - \left(\frac{b}{x}\right)^{\alpha}, b \leq x \qquad (4.15)$$

Where $1 < \alpha < 2.$ is the shape parameter and b is called the L*ocation Parameter*. The mean is given by $\alpha b/(\alpha - 1)$.

The main characterstics of a random variable obeying a heavy tailed distribution is that it exhibits *Extreme Variability*. Practically speaking, a heavy tailed distribution gives rise to very large values with non-negligible probability so that sampling from such a distribution results in the bulk of values being "Small" but a few samples having "very" large values. Not surprisingly, heavy tailedness impacts sampling by slowing down the convergence rate of the sample mean to the population mean, dilating it as the tail index α approaches *1*.

4.5.2 Heavy Tails and Long Range Dependence

Heavy-tailedness of certain Network-related variables -for example, file sizes and connection durations - can be shown to underlie the root *cause* of Long-Range Dependence and Self-Similarity in Network Traffic.

Now let us examine why Heavy Tails are considered the root cause of Long-Range Dependence in Network Traffic. Take a constructive approach by presenting input processes with probabilistic activity times, which then are shown to lead to Long-Rang Dependence if, and only if, they are heavy tailed. Let us present the *On/Off* model proposed by Willinger, et al[20].

The On/Off model considers N independent traffic sources $X_i(t), i \in [1, N]$, where each is a 0/1 *reward renewal* process with i.i.d. *On periods* and i.i.d. *Off-periods*. This just means that $X_i(t)$ takes on the values 1 ("***ON***") and 0 ("***OFF***") on alternating, non-overlapping time intervals called O*n and Off periods*; respectively. $X_i(t) = 1$ is interpreted as there being a packet transmission. Thus an On period can be viewed as constituting a "packet train". Three such on/off sources and their aggregation are depicted in Fig 4.4.

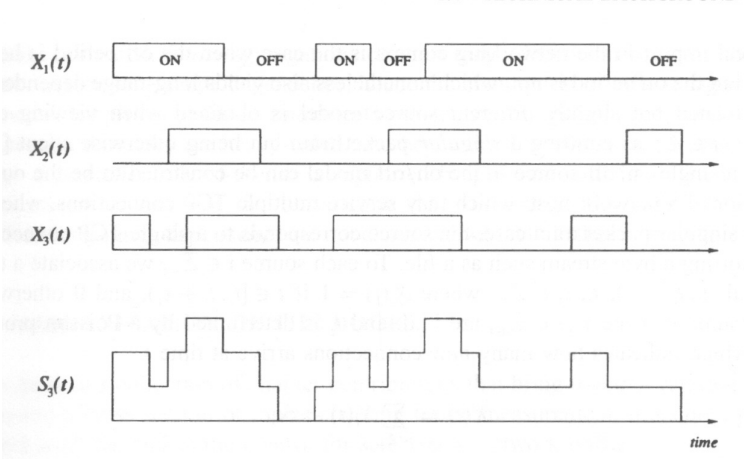


Fig. 4.4 N=3 On/Off Sources, $X_1(t)$, $X_2(t)$ and $X_3(t)$ and their aggregation $S_3(t) = X_1(t) + X_2(t) + X_3(t)$.

Let $S_N(t) = \sum_{i=1}^{N} X_i(t)$ denote the aggregate traffic at time *t*.

Consider the *Cumulative* process $Y_N(Tt)$ defined as

$$Y_N(Tt) = \int_o^{Tt} \left(\sum_{i=1}^{N} X_i(s) \right) ds \qquad (4.16)$$

Where T>0 is a scale factor that is explicitly incorporated. Thus $Y_N(Tt)$ measures the total traffic up to time Tt.

What is the behaviour of $Y_N(Tt)$ for Large T and N*?* Have the discussion simplified so as to concentrate on the single salient feature of how heavy tailedness influences Long Range Dependence.

Let τ_{on} be the random variable describing the duration of the *On* periods and let τ_{off} be the random variable associated with the durations of the *Off* periods.

Let

$$\Pr\{\tau_{on} > x\} \sim Cx^{-\alpha}, x \to \infty \qquad (4.17)$$

Where $1 < \alpha < 2$ and C>0 is a constant. As to τ_{off}, it can be either heavy tailed or light tailed with finite variance.

It can be shown that [20] $Y_N(Tt)$ behaves like Fractional Brownian Motion (FBM) as described in *Theorem 4.1.*

Definition 4.3. (*FBM*) Y(t), $t \in R$ is called ***Fractional Brownian Motion*** with parameter *H, 0<H<1,* if Y(t) is Gaussian and H-SSSi.

Definition 4.4. (FGN) X(t), $t \in$ Z+, is called ***Fractional Gaussian Noise*** with parameter *H* if X(t) is the *increment* process of FBM with parameter H.

Theorem 4.1. (On/off Model and FBM) $Y_N(Tt)$ behaves statistically as

$$\frac{E(\tau_{on})}{E(\tau_{on}) + (\tau_{off})} NTt + cN^{1/2} T^H B_H(t) \qquad (4.18)$$

for large T,N where $H=(3-\alpha)/2, B_H(t)$ is FBM with parameter H, and *c>0* is a quantity depending only on the distributions of τ_{on} and τ_{off}.

Thus $Y_N(Tt)$ asymptotically behaves as fractional Brownian motion fluctuating around $\frac{E(\tau_{on})}{E(\tau_{on})+(\tau_{off})}NTt$ when suitably normalized. It is Long-range dependent (1/2<H<1) if, and only if $1<\alpha<2;$ that is, τ_{on}'s distribution is heavy tailed. If *neither* τ_{on} nor τ_{off} is Heavy tailed, then $Y_N(Tt)$ is short-range dependent. It is this sense that heavy tailedness (in this case, of the On or Off periods) is an essential component to introducing Long-Range Dependence in the *aggregated* time series.

Empirical traffic measurements provide strong evidence that file sizes and connection durations are heavy tailed. And hence the heavy-tailedness causes Long-Range Dependence rule of thumb is supported by physical modeling.

4.6 Estimation of Long Memory

Since slowly decaying variance, Long-Range dependence and Spectral Density obeying a power law are different manifestations of second order self-similarity one can approach the problem of testing for and estimating self-similarity from *three* different angles: *(1)* Time domain analysis based on the R/S statistics *(2)* Analysis of the variance of the aggregated process and *(3)* the Periodogram based analysis in the frequency domain. Many methods for estimating the *self-similarity parameter H or the intensity of Long-range dependence* in a time series are available. They are typically validated by appealing to self-similarity or to an asymptotic analysis where one *supposes* that the *sample size* of the time series *converges to infinity*.

For the analysis of Internet Traffic and Packet Round Trip Delay time series data, I use two *Time domain* and one *Periodogram* based methods. Out of several Time domain methods to estimate the Long memory parameter *H,* the R/S statistic, which was first proposed by Hurst (1951) in a Hydrological context, and the Variance Plot are used. The Periodogram-based approach used in the analysis is Whittle Estimator.

4.6.1 R/S Statistic

Since ancient times, the Nile River has been known for its characteristics of Long Memory behavior. Long periods of dryness were followed by long periods of yearly returning floods. The famous hydrologist *Hurst* noticed these characteristics when he was investigating the question of how to regularize the flow of the Nile River. [7]

More specifically, his discovery can be described as follows: Suppose we want to calculate the capacity of a reservoir such that it is *ideal* for the time span between t and $t+k$. To simplify matters, assume that *time is discrete* and that there are *no storage losses*. By ideal capacity we mean that we want to achieve the following: *the outflow is uniform, that at time* $t+k$ *the reservoir is as full as at time* t*, and that the reservoir never overflows.*

Let X_i denote the inflow at time i and $Y_j = \sum_{i=1}^{j} X_i$ is the *cumulative* inflow up to time j.

Then the ideal capacity can be shown to be equal to

$$R(t,k) = \max_{0\le i\le k}\left[Y_{t+i} - Y_t - \frac{i}{k}\left(Y_{t+k} - Y_t\right)\right] - \min_{0\le i\le k}\left[Y_{t+i} - Y_t - \frac{i}{k}(Y_{t+k} - Y_t\right] \tag{4.19}$$

$R(t,k)$ is called the *adjusted range*. In order to study the properties that are independent of the scale, $R(t,k)$ is standardized by

$$S(t,k) = \sqrt{k^{-1}\sum_{i=t+1}^{t+k}(X_i - \overline{X}_{t,k})^2} \tag{4.20}$$

$$\text{where } \overline{X_{t,k}} = k^{-1}\sum_{i=t+1}^{t+k} X_i \tag{4.21}$$

$$\text{Then, the ratio } \frac{R}{S} = \frac{R(t,k)}{s(t,k)} \tag{4.22}$$

is called the *Rescaled Adjusted Range or R/S statistic.* Hurst plotted the logarithm of R/S against several values of *k*. He observed that, for large values of k, $\log R/S$ was scattered

around a *straight line* with a slope that exceeds ½. In probabilistic terminology this means that for large k,

$$\log E[R/S] \approx a + H \log k \text{, with } H > 1/2 \tag{4.23}$$

This empirical finding was in *contradiction* to results for Markov Processes, Mixing Processes, and other stochastic processes that where usually considered at that time.

For any stationary process with Short-range dependence, R/S should behave asymptotically like constant times $k^{1/2}$. Therefore, for large values of k, $\log R/S$ should be randomly scattered around a straight line with slope ½.

Hurst's finding that for the Nile River data, R/S behaves like a constant times k^H for $H > 1/2$, is known under the name *Hurst Effect*. Mandelbrot and Co-workers showed that the Hurst effect could be modeled by so called fractional Gaussian Noise with Self-similarity parameter $H \in (1/2,1)$.

With $Q(t,k) = \frac{R(t,k)}{S(t,k)}$, the R/S method can be summarized as follows:

1. Calculate Q for all possible (or for a sufficient number of different) values of t and k.
2. Plot $\log Q$ against $\log k$.
3. Draw a straight line $\log Q = a + b \log k$ that corresponds to the "ultimate" behaviour of the data. The coefficients a and b can be estimated, for instance by *least squares*. Set the value of H to be the estimated slope b of the fitted straight line.

4.6.2 Variance Plot

As discussed in section 4.4.2, one of the striking properties of Long-memory processes is that the variance of the sample mean converges slower to zero than n^{-1}, where n is the sample size. Formulating Eq (4.9) differently, one can have

$$Var(\overline{X}_n) \approx cm^{2H-2} \tag{4.24}$$

where c>0.

Equation(4.24) suggests the following method for estimating H:

1. Let k be an integer. For different Integers k in the range $2 \le k \le n/2$, and a sufficient number (say m_k) of sub series of length k, calculate the sample means $\overline{X}_1(k), \overline{X}_2(k), ..., \overline{X}_{m_k}(k)$ and the overall mean

$$\overline{X}(k) = m_k^{-1} \sum_{j=1}^{m_k} \overline{X}_j(k) \qquad (4.25)$$

2. For each k, calculate the sample variance of the sample means $\overline{X}_j(k), \quad j = 1,2,.....,m_k$.

$$S^2(k) = (mk-1)^{-1} \sum_{k=1}^{mk} (\overline{X_j}(k) - \overline{X}(k))^2 \qquad (4.26)$$

3. Plot $\log S^2(k)$ against $\log k$

For large values of *k*, the points in this plot are expected to be scattered around a *straight* line *with negative slope 2H-2*. The straight line is fitted with *least squares* method.

In the case of Short-range dependence or Independence, the ultimate slope is $2 \times 1/2 - 2 = -1$. Thus, the slope is steeper (more negative) for short memory processes.

4.6.3 Periodogram method and Whittle Estimator

The Periodogram of the time series:

$$I(\lambda) = \frac{1}{2\pi N}\left|\sum_{j=1}^{N} X_j e^{ij\lambda}\right|^2 \qquad (4.27)$$

where λ is a frequency, *N* is the number of terms in the series, and X_j is the data.

Because $I(\lambda)$ is an estimator of the Spectral density, a series with Long-Range dependence should have a Periodogram which is proportional to $|\lambda|^{1-2H}$ close to the origin. Therefore, a regression of the logarithm of the Periodogram on the Logarithm of the frequency λ should give a coefficient of $1-2H$. This provides an approximation to the parameter *H*.

The Whittle estimator is based on the Periodogram. It involves the function:

$$Q(\eta) = \int_{-\pi}^{\pi} \frac{I(\lambda)}{f(\lambda;\eta)} d\lambda \qquad (4.28)$$

where $I(\lambda)$ is the Periodogram and $f(\lambda;\eta)$ is the spectral density at frequency λ, and where η denotes *the Vector of Unknown Parameters.* The Whittle Estimator is the value of η which

Minimizes the function $Q(\eta)$. When dealing with Fractional Gaussian Noise or Fractional ARIMA, η is simply the parameter H or D. [7]

4.7 Summary

This section *summarizes* the properties of the collected Internet Traffic and Packet Round Trip Delay time series, as well as the results of the Hurst Parameter estimations applied on the collected data. The analyses based one the graphical and mathematical tools convincingly establish the *self-similar* nature of Internet packet traffic and round trip delay processes.

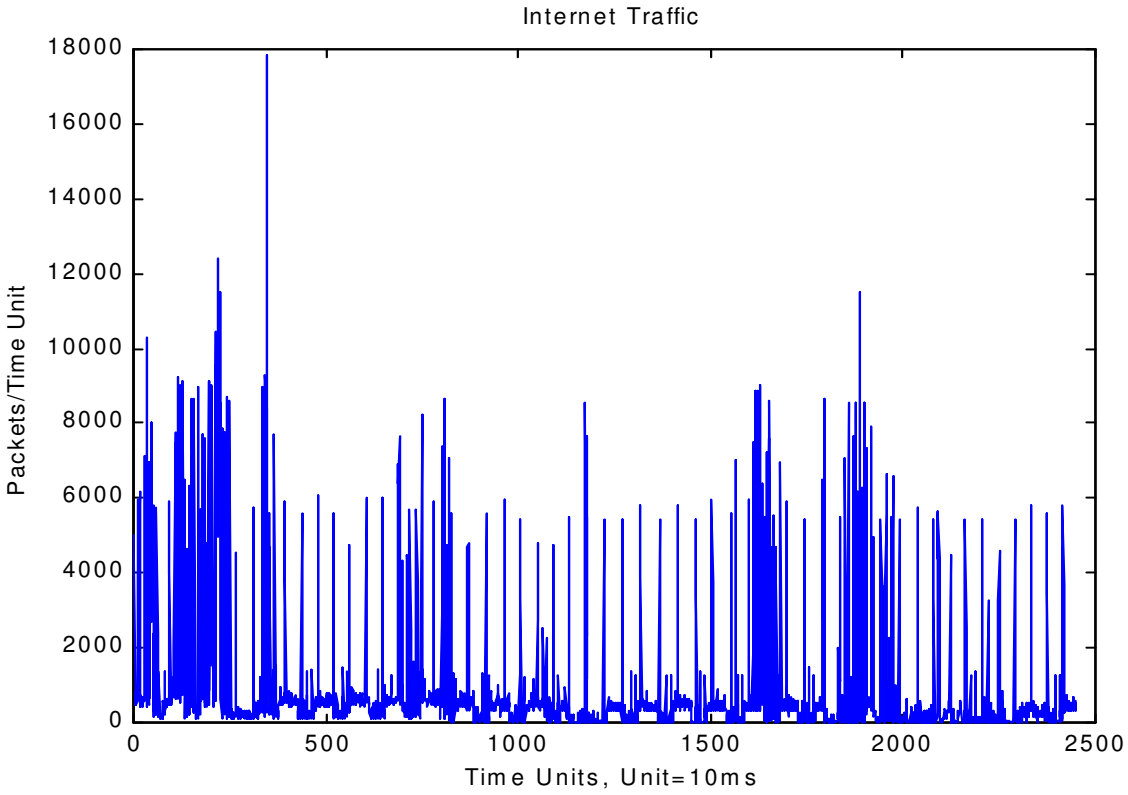
4.7.1 Self-Similarity of Internet Traffic

This section presents the analysis results done to track the *statistical properties* of the collected *Internet Traffic* data. The collected traffic data is a count of the number of packets or bytes traversing a selected port per a *10ms* time unit. The collection is done using *AG Group Etherpeek version 3.5* for trapping packets at the Internet Gateway in a company Local Area Network (*Icon Networks)* and Artiza Packet Trapping Software for trapping Internet packets at an Internet Service Provider Gateway (ETC). These software tools only count the traffic *Intensity (number of Packets)* and have no facility for revealing the content of the packets. The effort to trap Traffic at AAU's Internet Gateway was unsuccessful due to lack of Software to run on a Sun Solaris System.

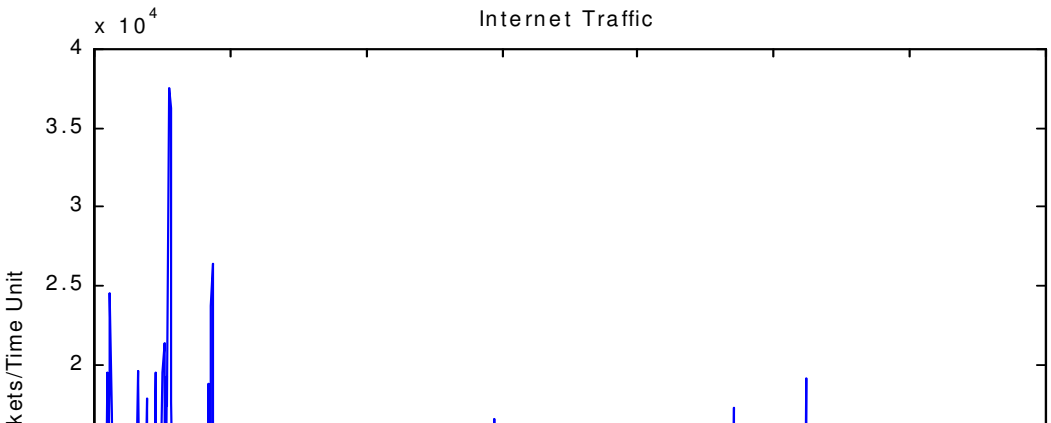
4.7.1.1 Scale Invariance and Autocorrelation

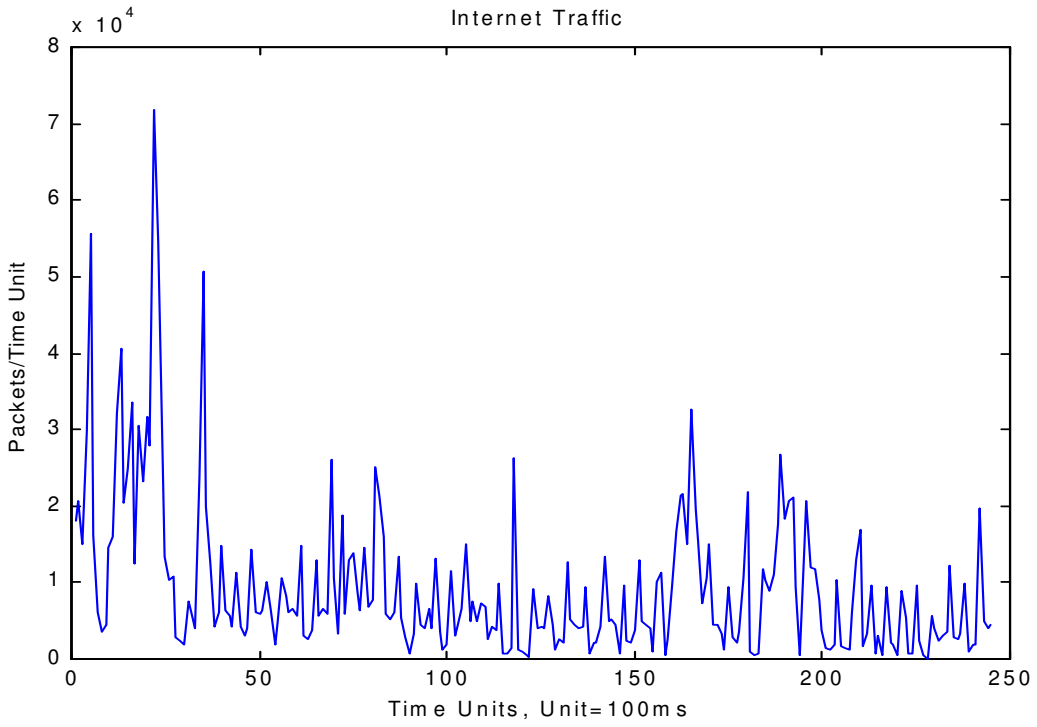
The minimum resolution of the packet trapping software is 10ms. The number of bytes of the packets collected for each 10ms time interval is sequenced as a *time series.* Then these time series are analyzed to investigate the statistical properties.

Figure 4.6 (*a, b, and c)* show us the plot of the *Packet Traffic* in *Bytes per time scale* versus index of the sequence in three time units. In Figure 4.6 *(a)* the original time series is displayed (that is, with a time unit of 10ms). In Figure 4.6 *(b)* the original time series is aggregated into 4 levels (that is, with a time unit of 40ms). In Figure 4.6 *(c)* the original time series is aggregated into 10 levels (that is, with a time unit of 100ms).



(a)



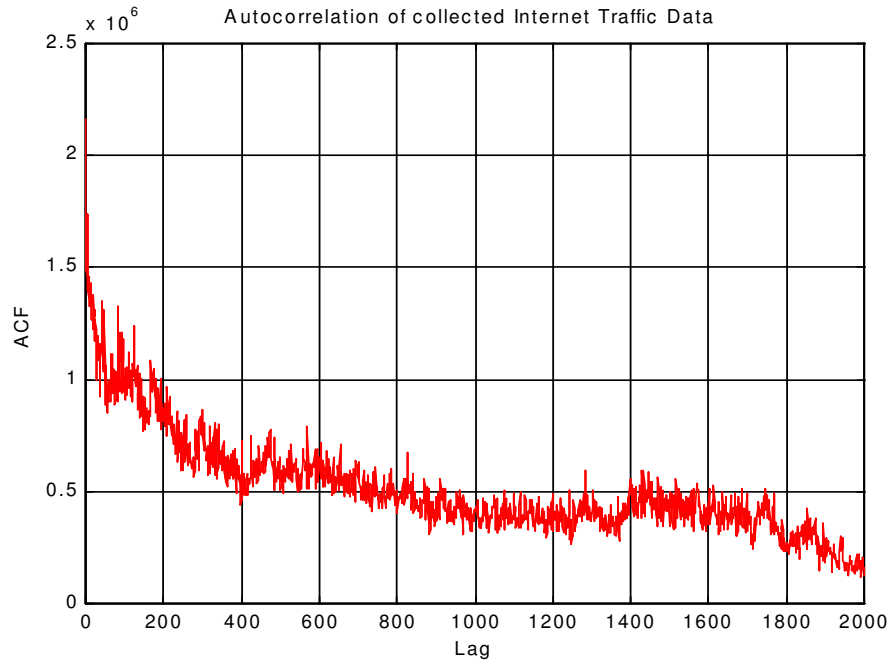


(c)

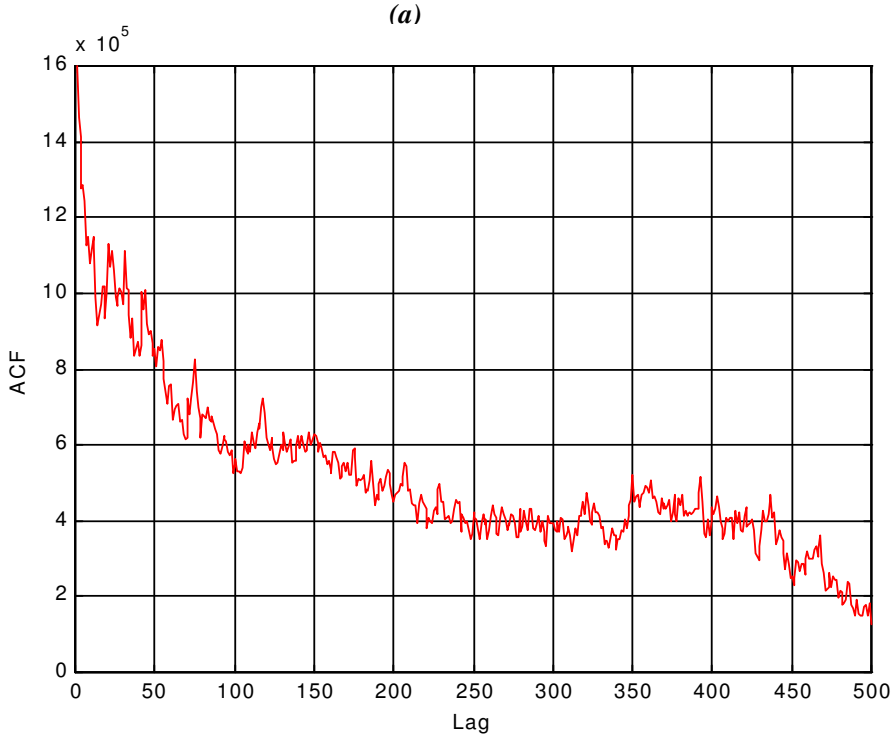
Fig 4.6 Packet Traffic Traces for different time scales (a) 10ms (b)40ms and (c) 100ms

As can be seen from Figure 4.6, the *burstiness* of the traffic never *smoothed* even though different levels of aggregation are used. This suggests there is no natural length of a burst in Internet traffic; the bursts remain in all levels of aggregation. Intuitively the three plots look very similar to one another (in a distribution sense) and are distinctively different from white noise (i.e., an independently, identically distributed (iid) sequence of random variables). This is the property of *Self-Similar Processes***.**

Figures 4.7 *(a) and (b)* depict the Autocorrelation of the original time series (10ms) for the first 2000 lags and its aggregated (50ms) time series for the first 500 lags, respectively.



(a)



(b)

Fig 4.7 Autocorrelation of (a) the original time series (b)its 50ms

As can be seen from Figure 4.7, the Autocorrelation decays *slower* than the exponential rate. It is unlike Short-Range dependent data series that are shown in Figure 4.8 *(which is depicted from a synthesized ARMA (2,2) series),* included for comparison purposes.

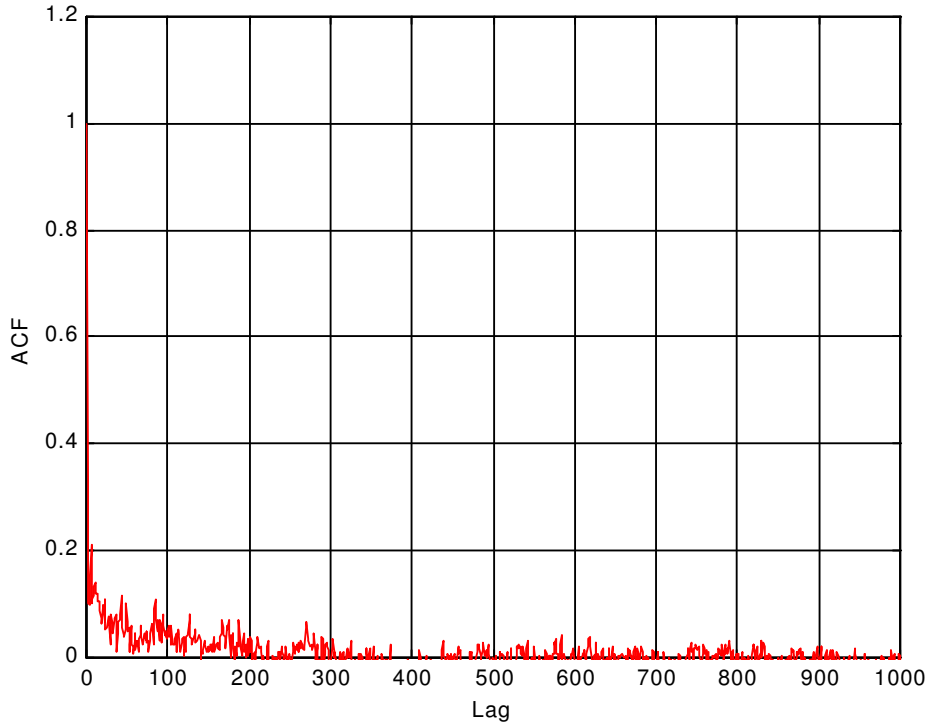


Fig 4.8. Typical Short Range Dependence Autocorrelation :obtained from a synthesis of ARMA(2,2) process.

As discussed in section 4.4 and 4.6, the *intensity of Self-Similarity and Long-Range Dependence* can be catched through the help of the *Hurst Parameter.* The following section presents the results of the Hurst Parameter estimation techniques applied to the collected Internet Traffic Data. (*Recall that values of H in the range (0.5 < H <1) imply Self-Similarity or Long-Range dependence).*

4.7.1.2 Hurst Parameter Estimation

R/S analysis

The result obtained after analyzing the collected Internet Traffic data using the *Rescaled Range (R/S)* analysis shows the self-similar behavior of the Internet traffic. The value of *H* (Hurst parameter) obtained was ***0.90126***.

Figure 4.9 depicts the log-log plot of R/S statistic. It shows an asymptotic slope that is distinctly different from 0.5 and is estimated using least squares to be 0.90126.

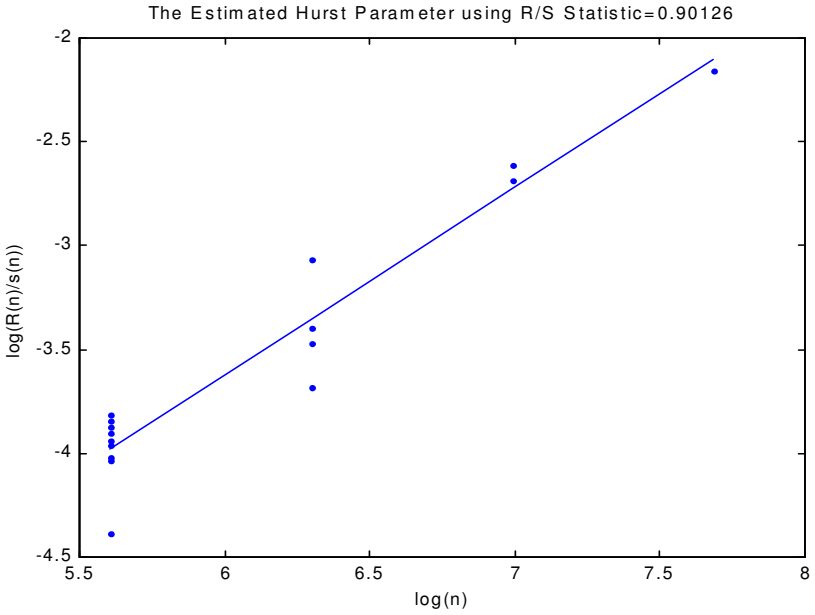
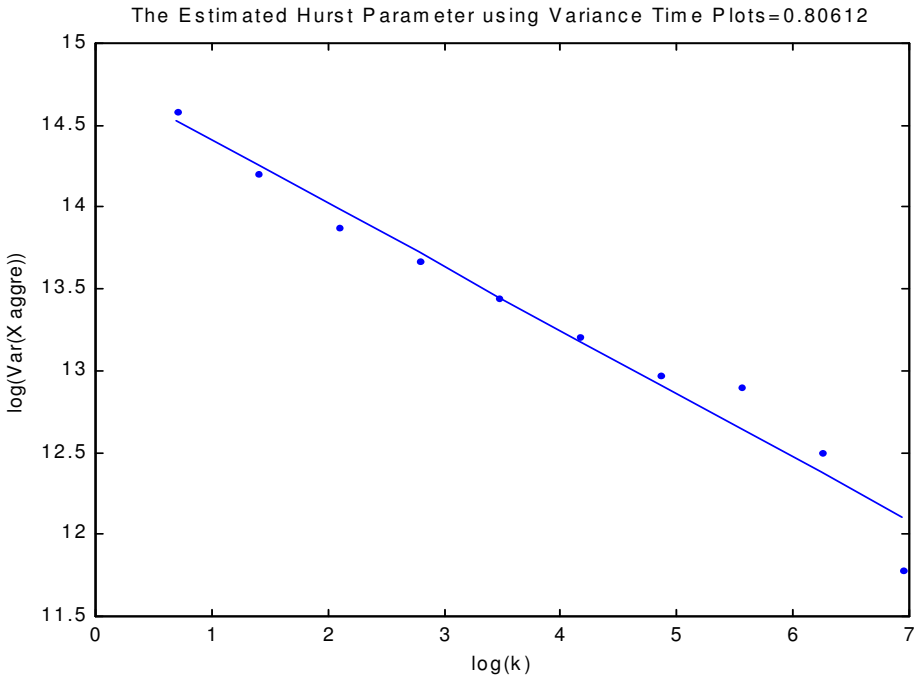


Fig. 4.9. R/S Statistic Hurst parameter estimation

VARIANCE PLOT

The result obtained using the Variance Plot method proves that the data has long memory or it is long-range dependence. The estimated Hurst parameter equals ***0.80612***. Figure 4.10 asserts this.



The variance-time curve shows an asymptotic slope that is clearly different from –1 and estimated to be -0.3878 (using least squares), resulting in an estimate of the Hurst parameter *H* of about 0.80612.

THE WHITTLE ESTIMATOR

The result obtained using the Whittle Estimator method proves that the data has long memory or it is long-range dependence. The estimated Hurst parameter is between *0.96 and 0.97*.

4.7.1.3 Conclusion

From the above observations, we can get the following conclusions on the nature of Internet packet traffic:

- The bursts in the traffic observed in all levels of aggregation; and not smoothed by having a larger time unit of observation, unlike *short-memory* processes such as the

Poisson distribution. The data sets look similar in the distribution sense. Hence, we can decide that *Internet traffic is statistically Self-Similar.* The autocorrelation function doesn't decay exponentially, a slow decay is observed. Hence the data set has *Long Memory, or is Long Range Dependence* (LRD). The degree of self-similarity measured in terms of the Hurst parameter H tends to be close to 1, indicating intensive self-similarity.

- The long-range dependence and scale invariant "burstiness" of the Internet traffic is drastically different from both conventional telephone traffic and from stochastic models for packet traffic usually considered in the literature (Example is the *Poisson* Model). Traffic models for voice traffic, developed over the years to serve the telephone network, did not apply as might have been hoped because voice traffic does not give rise to the same traffic characteristics as Internet data traffic, which is *burstier*.

4.7.2 Self-Similarity of the Packet Round Trip Delay Process

A packet round-trip delay is the sum of delays on each subnet link traversed by the packet, from the time the packet originates from the sender until it receives back at its origin. Each link (or hop) in turn consists of *four* components, including *Processing* delay, *Queueing* delay, *Transmission* delay and *Propagation* delay. So a packet Round Trip delay ***process,*** $T(t)$,is a *random variable* at time t. $T(t)$ describes the *packet round trip delay process*. (*Section 6.2 will present detailed information and further statistical investigations on Packet Round Trip Delay Processes)*

In this section I present the characterstics of the time series T_i. It is obtained by *discretizing* $T(t)$ with *t*. Simply, T_i is a *sample process* of $T(t)$ at $t = t_i$ where $i = 1,2,3,.....$In this context t_i is the time at which the ping echo reply is received.

I used a custom application developed in Microsoft Visual Basic, named *VBPing* (© Mal Maliska, a *freeware* software downloaded from the Internet), that uses TCP/IP's, *Ping* application to calculate and record packet round trip delay. The recorded packet Round Trip Delays or Round Trip Times (RTT) will be used in the statistical analysis.

The ping client is a Workstation in our office (*ICON Networks)* and the Ping server is *yahoo.com (IP Address: 64.58.79.230). (I will consider additional Ping Server in Section 6.2)*. I tried to collect data with as many instances as possible. The ping interval is *5 Seconds.*

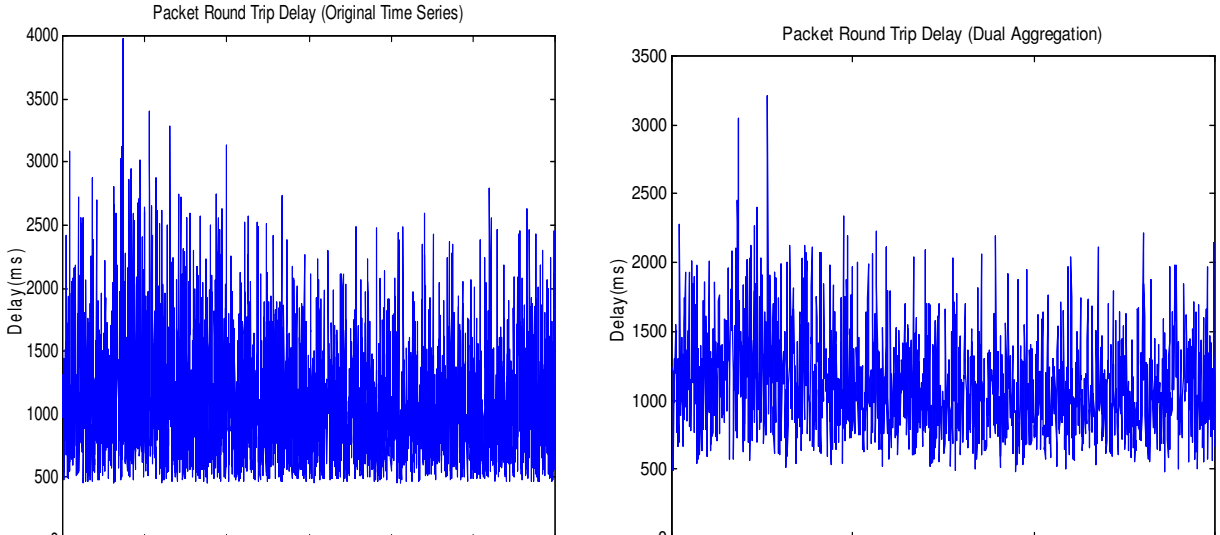
The Round Trip Times measured in millisecond get collected and arranged in a time series and the statistical behaviors of the resulted sequence studied.

4.7.2.1 Scale Invariance and Autocorrelation

Figure 4.11 (*a, b, c, and d)* show us the plot of the *Packet Traffic* versus time in four levels of aggregation (averaging). In Figure 4.6 *(a)* the original sequence is displayed. In Figure 4.6 *(b), (c) and (d*) in dual, quadruple and fifth aggregation, respectively.

Intuitively the four plots look very similar to one another. As can be seen from the Figures, the *burstiness* of the delays never *smoothed* even though different levels of aggregation were used. See that there exists a "burst" like traffic in all time scales. It depicts a high variability in Packet Round Trip Delays in the Internet.

Figure 4.12 depict the Autocorrelation of the collected Packet Round Trip Delay sequence for the first 2000 lags. As can be seen, the Autocorrelation decays *slower* than the exponential rate.



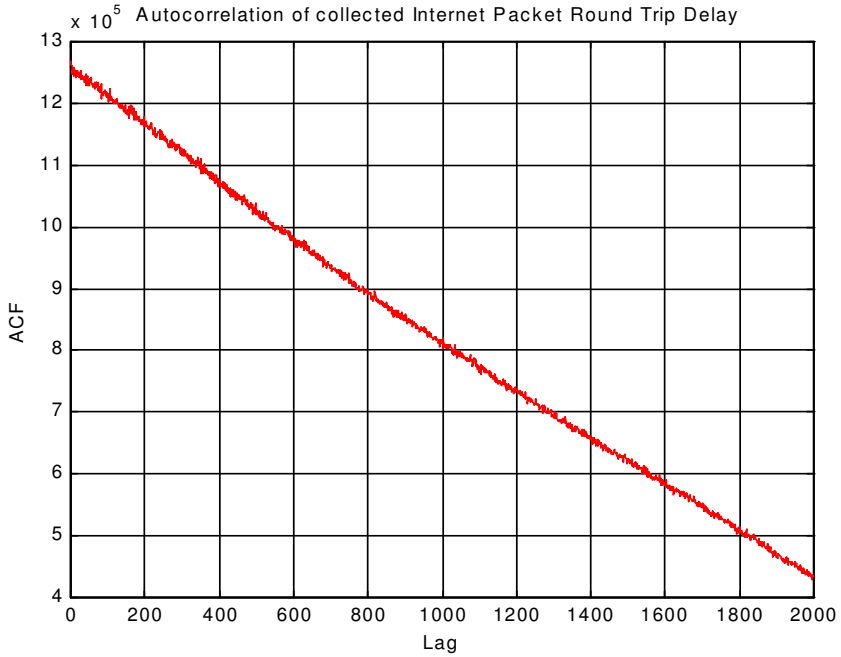


Fig. 4.12. Autocorrelation the Packet Round Trip Delay Sequence.

4.7.2.2 Hurst Parameter Estimation

R/S analysis

The result obtained after analyzing the collected data using the *Rescaled Range (R/S)* analysis shows the self-similar behavior of the Internet traffic. The value of *H* (Hurst parameter) obtained was ***0.81822***.

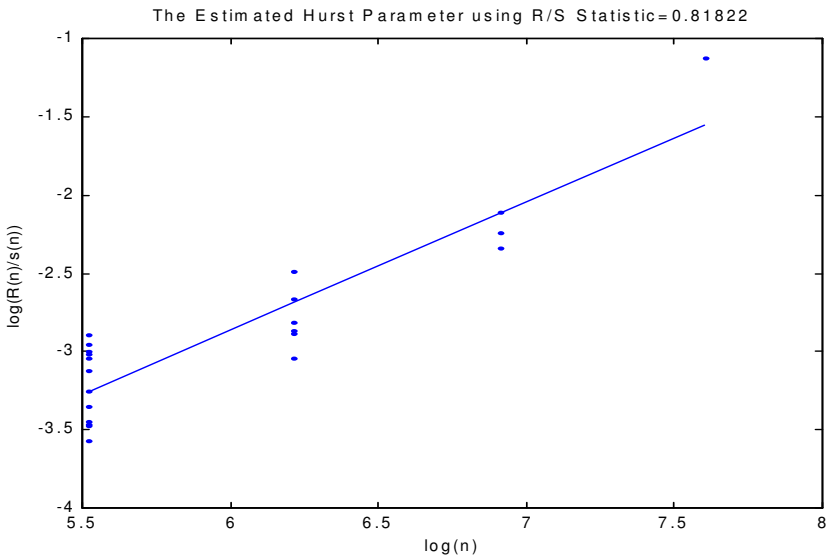


Fig 4.13 R/S analysis of the Packet Round Trip Delay sequence

VARIANCE PLOT

The result obtained using the Variance Plot method proves that the data has long memory or it is long-range dependence. The estimated Hurst parameter equals ***0.80346.***

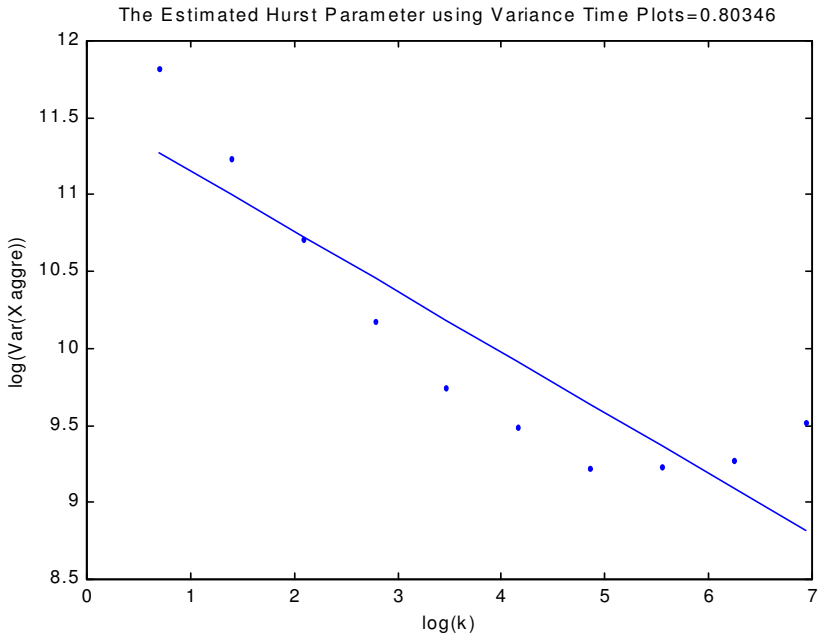


Fig. 4.14. Hurst Parameter estimation using Variance Time plot

3.7.2.3 Conclusion

From the above observations, the following conclusion is made.

- The bursts in the Packet Round Trip Delays are observed scale-invariantly, they exist in all levels of aggregation; and not smoothed by having a larger time unit of observation, unlike *short-memory* processes such as the *Poisson distribution.* The data sets look similar in the distribution sense. Hence, we can decide that *Packet Round Trip Delay process is statistically Self-Similar.* The autocorrelation function doesn't decay exponentially, a slow decay is observed. Hence the data set of the Packet Round Trip Delay process has *Long Memory, or is Long Range Dependence* (LRD).

CHAPTER 5

Modelling Self-Similar Network Traffic

5.1 Introduction

High-speed Network traffic mechanisms are propelling the growth of a new class of communication services such as *multimedia*, *video on demand*, etc. As these new communication services evolve and the needs of the users change, existing communication backbone needs to be modified or replaced by completely newer systems. Telecommunication professionals are being called up on to the design and mange these systems in the face of fast moving technology and a climate of increasing customer expectations. Towards this end, B-ISDN (Broadband-Integrated Services over Digital Networks) networks now attempt to provide a guarantee of the quality of service (QoS) deliverable to the customers.

Design and management decisions are, to the major part, based on models that predict the performance of network. Analytical techniques, computer simulations, projections from existing experience and experimentation are methods used to evaluate and compare network design and protocol. [9].

Performance models require *simulation* of traffic using models that can accurately capture the statistical characteristics (*Self-similarity or Long-Range Dependence)* of the actual traffic. If the models do not accurately represent the traffic, then one may *overestimate* or *underestimates* network performance. Decisions based on poor predictions may adversely affect the customer's perception of new technology.

Traditional network traffic models are based on the assumption that traces exhibit *short memory* only. These models include *Renewal Theory, Markov Chains* (with several variations) etc. The presence of Long-Memory was first detected in the Ethernet Measurements at Bellcore. [1] Since then several measurements over networks including *LAN, WAN, ATM*, and traffic generated by commonly used applications such as *TELNET,FTP,WWW*-----) were collected and analyzed. As demonstrated in Chapter 4, the experiments clearly demonstrate the presence of Long-Memory in Traffic traces. This discovery mandates the use of models that can catch long-memory.

From the experimentations done, *FARIMA* (*Fractional Auto Regressive Integrated Moving Average)* models proved to be a better choice of the available models. FARIMA model offers flexibility in modeling *both* short memory and long memory characteristics. FARIMA process is an extension of ARMA process and it represents parsimoniously, stationary time series that

exhibit short-memory and long-memory. Alternatively, FARIMA processes can also be viewed as an incremental processes obtained from a statistically Self-Similar non-stationery process.

The following sections present descriptions and mathematical analysis on the *candidate* models considered for representing self-similar processes. Section 5.5 discusses the model selection criteria for representing self-similar processes as well as the synthesis of self-similar traces through the implementation of the selected model.

5.2 ARMA processes

ARMA (*Auto Regressive Moving Average*) processes represent and important class of series $[X(t): t \in z]$ defined interms of linear difference equations with *constant coefficients*.

The general form of and ARMA (p, q) process is written as

$$\alpha(\beta)X(t) = \beta(B)Z(t) \qquad (5.1)$$

where $Z(t)$ is the independent identically distributed (i.i.d) process, $\alpha(B)$ *and* $\beta(B)$ are the p^{th} and q^{th} degree polynomials, expanded as:

$$\alpha(B) = 1 - a(1)B^1 - a(2)B^2 - \text{----------} - a(p)B^p \qquad (5.2a)$$

$$\beta(B) = 1 + b(1)B^1 + b(2)B^2 + \text{----------} + b(q)B^q \qquad (5.2b)$$

representing the AR(Auto Regressive) and MA(Moving Average) components respectively, and B is the backward shift operator defined by $B^j X(t) = X(t-j)$.

Denote the autocorrelation function of $X(t)$ as $r_x(\tau) = E[X(t)X(t+\tau)]$. *Traditional* time series analysis assume that $X(t)$ is "*Mixing* "i.e.

$$\sum |r_x(\tau)| < \infty \qquad (5.3)$$

and hence $r_x(\tau)$ decays *exponentially* as τ increases, which implies that samples of $X(t)$ that are well separated in time are approximately uncorrelated. Due to this behavior of the Autocorrelation structure, ARMA processes are also referred to as *Short-Memory* or *Short-Range dependent* processes.

5.3 ARIMA processes

An *ARIMA (Auto Regressive Integrated Moving Average)* process is used to represent a class of *non-stationary* time series $\{Y(t) : t \in \mathrm{Z}\}$ that exhibit, homogeneity in that apart from their local level and/or trend, one part of the series behaves like any other part [9]. Figure 5.1 shows such trends.

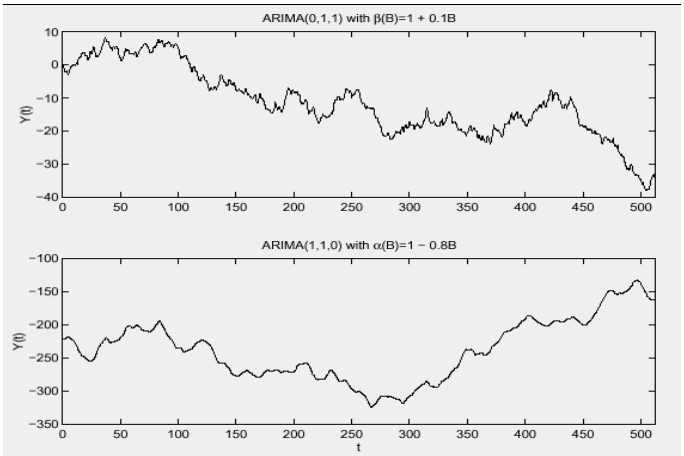


Fig. 5.1. ARIMA(0,1,1) process (Top) and ARIMA(1,1,0) process (bottom)

In other words, if the variations in the local level and/or trend are removed, the series becomes stationary. Such series are represented by a generalized *autoregressive operator* $\phi(B)$ in which one or more roots of the equation $\phi(B)$ is unity

The operator $\phi(B)$ is written as

$$\phi(B) = \alpha(B)(1-B)^d \qquad (5.4)$$

where d is the multiplicity of the root at $B = 1$ and is an integer. And $\alpha(B)$ *and* B represent same roles as their definitions in ARMA processes. Thus a general model, which can represent homogenous non-stationary behavior, is of the form:

(5.5a)

$$\phi(B)Y(t) = \beta(B)z(t)$$

$$or\ \alpha(B)(1-B)^d Y(t) = \beta(B)z(t) \qquad (5.5b)$$

$$or\ \alpha(B)X(t) = \beta(B)z(t) \qquad (5.5c)$$

$$where\ \ X(t) = (1-B)^d Y(t) \qquad (5.5d)$$

The reason for including the term "Integrated" in the ARIMA title is due to the following relationship, which is in fact the inverse of Eq (5.5d)

$$Y(t) = S^d X(t) \qquad (5.6)$$

where S is a *summation (*or in the case of continuous functions *–Integration)* operator, defined by:

$$S = \frac{1}{1-B} = \sum_{j=0}^{\infty} B^j \qquad (5.7)$$

Thus, a general ARIMA process can be generated from an innovation process $Z(t)$ by means of three *filtering* operations as indicated in Fig 5.2.

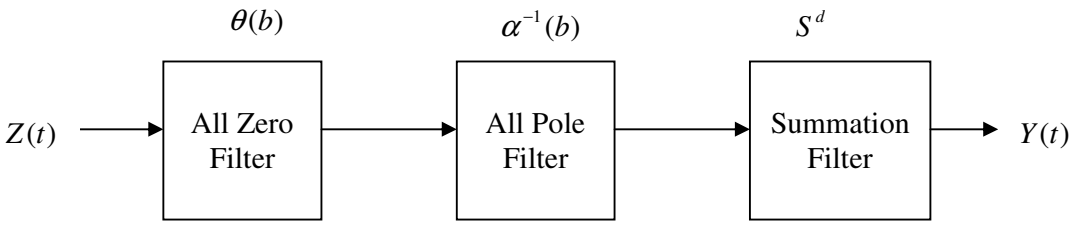


Fig. 5.2. Block Diagram representing synthesis

5.4 FARIMA processes

An ARIMA process can be considered as a first step towards the generalization of ARMA processes. In ARIMA processes the parameter d assumes *integer* values only. FARIMA processes are obtained by relaxing this constant, i.e by permitting d to take fractional values.

Definition 5.1. Let $X(t)$ be a stationary process such that

$$\alpha(B)(1-B)^d X(t) = \beta(B)z(t) \qquad (5.8)$$

for some $d \in (-0.5,0.5)$. Then $X(t)$ is a ***FARIMA (p,d,q)*** process, where p and q represent the degree of the AR and MA polynomials defined in ARMA process.

Since $d \in (-0.5,0.5)$ this implies that $X(t)$ has fractional pole at $B=1$. The upper bound $d < 0.5$ is needed because for $d > 0.5$ the process is *not stationary*. However, the case $d < 0.5$ can be reduced to the case $-0.5 < d \le 0.5$ by taking the appropriate differences. For example, if equation (5.8) holds with d=1.2, then the differenced process $X(t) - X(t)$ is its solution with d=0.2. If $d = \pm 0.5$, then $X(t)$ is either stationary or invertible but not both.

When $0 < d < 0.5$, *FARIMA (p,d,q)* process *exhibits Long-memory*. Parameters p and q allow for flexible modeling of *short-range* characteristics.

It must be noted that, although ARIMA & FARIMA processes have the same model, the former is non-stationary while the later is stationary process.

The ARMA parameters p and q define the correlation structure at *smaller lags*. On the other hand, the long memory parameter, *d,* affects the rate of decay of $r_x(\tau)$ as $\tau \to \infty$. The close d is to 0.5, the more significant is the long-memory.

The asymptotic decay of the autocorrelation function is such that

$$r_x(\tau) \sim C|\tau|^{2d-1} \qquad (5.9)$$

where $C \neq 0$ and $0 < d < 0.5$

The S*pectrum* of *FARIMA (p,d,q)* follows directly form Eq (5.8)

$$f_x(w) = \frac{\sigma_z^2}{2\pi}\left|1-e^{-jw}\right|^{-2d}\left|\frac{\beta(e^{jw})}{\alpha(e^{jw})}\right|^2 \qquad (5.10)$$

Since $\left|1-e^{-jw}\right| = 2\left|\sin(w/2)\right|$ and $\lim_{w\to 0}\frac{2\sin(w/2)}{w} = 1,$ the behavior of the spectral density as $w \to 0$, is given by

$$f_x(w) = \frac{\sigma_z^2}{2\pi}|w|^{-2d}\left|\frac{\beta(1)}{\alpha(1)}\right|^2 \qquad (5.11)$$

Equation (5.11) implies that when $0 < d < 0.5$, $f_x(w)$ is unbounded at $w = 0$, i.e.,

$$f_x(w)\,|_{w=0} = \infty \qquad (5.12)$$

$$\sum r_x(\tau) = \infty \qquad (5.13)$$

indicating *sub-exponential* decay of the autocorrelation function.

5.4.1 Parameter Estimation Techniques of FARIMA (p,d,q)

The parameter estimation problem for *FARIMA (p,d,q)* process involves estimation of d, which describes the long memory characteristics, and estimation of vectors $\boldsymbol{a}=[1, a(1),.....a(p)]$ and $\boldsymbol{b}=[1, b(1),.....b(q)]$ which describe the short memory characteristics. Existing estimation methods can be classified into *two* categories.

1. *Heuristic Methods:* methods that estimate d only. Then after removing the presence of Long-memory characterstics form the data one can estimate the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ using the traditional ARMA parameter estimation methods.

2. *Periodogram Methods:* methods that jointly estimates d, $\boldsymbol{a}$ and $\boldsymbol{b}$.

HEURISTIC METHODS

Heuristic methods were originally proposed to estimate the *Hurst parameter (H)* for self-similar processes. They can be used for estimating d from a FARIMA process $X(t)$ since $d = H - 0.5$ and the cumulative process $S(t)$, defined as $S(t) = \sum_{n=0}^{t} X(n)$, is self-similar as $t \to \infty$.

These methods include the Variance Plots and R/S Statistics, which were dealt in *Chapter 3*.

PERIODOGRAM METHODS

These methods are mostly used to jointly estimate d, $\boldsymbol{a}$ and $\boldsymbol{b}$. One of the popular methods, Whittle estimator has been discussed in Chapter 4. Other estimation methods that have been

proposed for estimating d includes *Higuchi's method*, *Localized Whittle's Method*, and *Wavelet based methods*.

5.4.2 Fitting a FARIMA model

The applicability of FARIMA models is dependent on the ease with which they may be fitted to an observed time series. Selection and estimation for FARIMA model is more difficult than that of standard ARMA models. When both long- and short-range correlation structures are present in the data, their behavior is difficult to distinguish. Additionally, likelihood estimation techniques for FARIMA model present computational problems and often result in significant finite sample biases. However, methods for constructing ARMA models are now well established. FARIMA model is the extended version of ARMA model in nature, so it will be helpful to solve model-building problems of FARIMA by applying the effective methods for ARMA processes.

One can transfer the FARIMA problem to the ARMA problem by splitting FARIMA model into its "*fractional differenced*" and "*ARMA*" parts. For a FARIMA process $X(t)$,

$$\phi(B)\Delta^d X(t) = \varphi(B_t)z(t) \qquad (5.14)$$

where $\Delta = (1-B)$

And obtain

$$W(t) = \Delta^d X(t)$$
$$\text{where } W(t) = \left[\phi(B)/\varphi(B)\right]z(t) \qquad (5.15)$$

So, if the differencing parameter d can be obtained in *advance* and the fractional differencing operator Δ^d can be implemented in practice, then *W(t)*, the fractionally differenced *X(t)*, can be evaluated as an ARMA model.

The following is the procedure used to ***fit*** a FARIMA model to the data.

Step 1. Estimating fractional differencing parameter *d*

Hurst parameter H can be estimated by several methods such as Variance-Plot analysis, *R/S* analysis and periodogram-based analysis. The strength of long-range dependence measured by d is the same as that by H upon $H = d + 0.5$. Then we can estimate parameter d from this relationship.

Step 2. Fractional differencing on *X(t)*

The calculation of *W(t)* from *X(t)* involves a *finite* approximation to the infinite sum in the definition of fractional differencing operator. The exact fractional differencing operator for the *X(t)* series with mean value μ is

$$\Delta^d (X_t - \mu) = \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k (X_t - \mu) = \sum_{j=0}^{\infty} \varpi_j (X_{t-j} - \mu) \qquad (5.16)$$

where

$$\varpi_j = \frac{(-1)^j \Gamma(1+d)}{\Gamma(1+j)\Gamma(1+d-j)}$$

This involves the unobserved quantities *X(0), X(-1),….* We have assumed ***causality*** here.

After using the operator Δ^d with estimated parameter *d* on *X(t)*, we can obtain *W(t)*.

Step 3. Model identification and parameter estimation for the FARIMA model.

Now we can analyze *W(t)* as an *ARMA(p,q)* model using conventional methods. Once the order *p* and *q* are *selected*, we can estimate all the parameters of the desired FARIMA model by *Durbin recursion* method. [12]

5.4.3 Synthesis of FARIMA (p,d,q) process

While a packet trapping software such as *Etherpeek* is useful for gathering trace data, not everyone has access to such software. To do simulation studies, a lot of trace data is needed, and gathering and storing this data is expensive both interms of time and storage requirements. There is also the question of whether users of the *Networks* feel comfortable having their network traffic monitored and recorded. What is needed is a fast method of generating synthetic traffic that resembles the actual traffic.

The goal of any FARIMA *synthesis* algorithm is to *generate sequences*, which exhibit Long-Memory asymptotically and be computationally attractive for generating long data records.

The synthesis procedure can be split into two tasks.

- Generate *FARIMA (0,d,0)* sequence exhibiting Long-Memory, and
- Add short memory to it with selected parameters *p and q*.

The *method* we used for synthesizing the *FARIMA(0,d,0)* processes is using *Hosking's Algorithm.* Hosking provides an algorithm for generating a long-range dependence process called *FARIMA(0,d,0)*, where the zeros indicate there are no Auto Regressive (AR) and

Moving Average(MA) parameters specified. The basic equations for Hosking's algorithm are as follows. [13]

The process X_k has Gaussian marginals with *zero* mean and variance v_o, and fractional differencing parameter $d = H - 1/2$. The autocorrelation function has an asymptotically hyperbolic shape, and is determined from d as

$$\rho_k = \frac{d(1+d)(2+d).....(k-1+d)}{(1-d)(2-d).....(k-d)} \qquad (5.17)$$

X_o is chosen from the Normal distribution $N(0, v_o)$. Set $N_o = 0$ and $D_o = 1$. Then generate n points by iterating the following for $k = 1,2.....n$:

$$N_k = \rho_k - \sum_{j=1}^{k-1} \phi_{k-1,j}\rho_{k-j} \qquad (5.18)$$

$$D_k = D_{k-1} - N_{k-1}{}^2 / D_{k-1} \qquad (5.19)$$

$$\phi_{kk} = N_k / D_k \qquad (5.20)$$

$$\phi_{kj} = \phi_{k-1,j} - \phi_{kk}\phi_{k-1,k-j} \qquad j = 1,....,k-1 \qquad (5.21)$$

$$m_k = \sum_{j=1}^{k} \phi_{kj} X_{k-j} \qquad (5.22)$$

$$v_k = (1 - \phi^2{}_{kk}) v_{k-1} \qquad (5.23)$$

Choosing each X_k from $N(m_k, v_k)$. Since each point depends on every previous point, this algorithm requires $O(n^2)$ computation time.

The second task is relatively easy and can be implemented as a ***filter*** operation. For example, if $\boldsymbol{a}=[1, a(1).....a(p)]$ and $\boldsymbol{b}=[1, b(1).....b(q)]$ represent the AR*(Auto Regressive)* and MA*(Moving Average)* coefficients, then the command *filter* (b, a, X) in ***MATLAB*** will add short-memory characteristics to data vector X, which represent FARIMA(0,d,0).

5.5 Summary

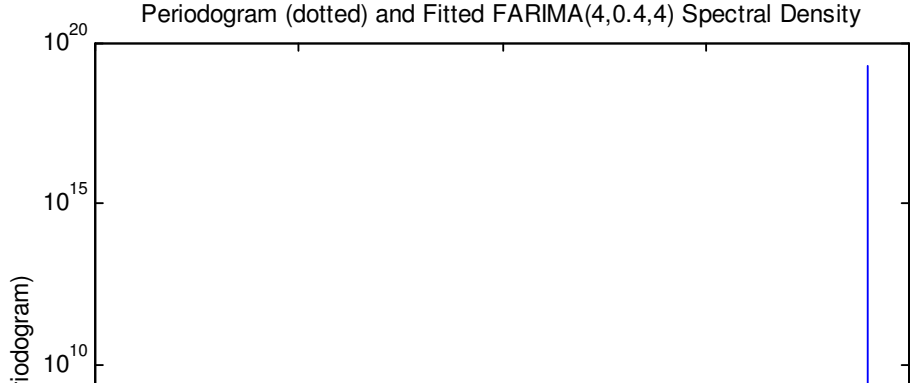
5.5.1 Model Selection.

Out of the models discussed in the previous sections, decisions made on a model that properly represent *Self-Similar processes.* This is done by *fitting* the *periodogram* of the collected *Internet Packet Traffic* (which has been proved to be self-similar in Chapter 4) to the *spectrum* of the fitted Model.

Figure 5.3 displays the result for the fitted *FARIMA(4,0.4,4)* model as plotted along with the periodogram of the collected Internet traffic. Figure 5.4 displays the result for the fitted ARMA(4,4) model as plotted along with the periodogram of the collected Internet traffic time series. Due to its *Non-Stationarity* nature, the ARIMA model is avoided from comparison.

It can be seen that the FARIMA model has fitted both the Long-Range *(Small Frequency)* and Short-Range *(High Frequency)* properties with the Periodogram *appropriately*. The conclusion drawn from this result is that FARIMA process is capable of *simultaneous* modeling of Long-Range and Short-Range behavior of Network Traffic. Purely long-memory models can be modeled by setting $p = q = 0$.

Many researchers argue that Network Traffic is better modeled by *Fractional Gaussian Noise (FGN), but* as been confirmed *FARIMA* model can be more useful since, besides modelling Long-Memory, it provides flexibility in modelling Short-Memory properties. *FGN* only catches Long-Memory properties.



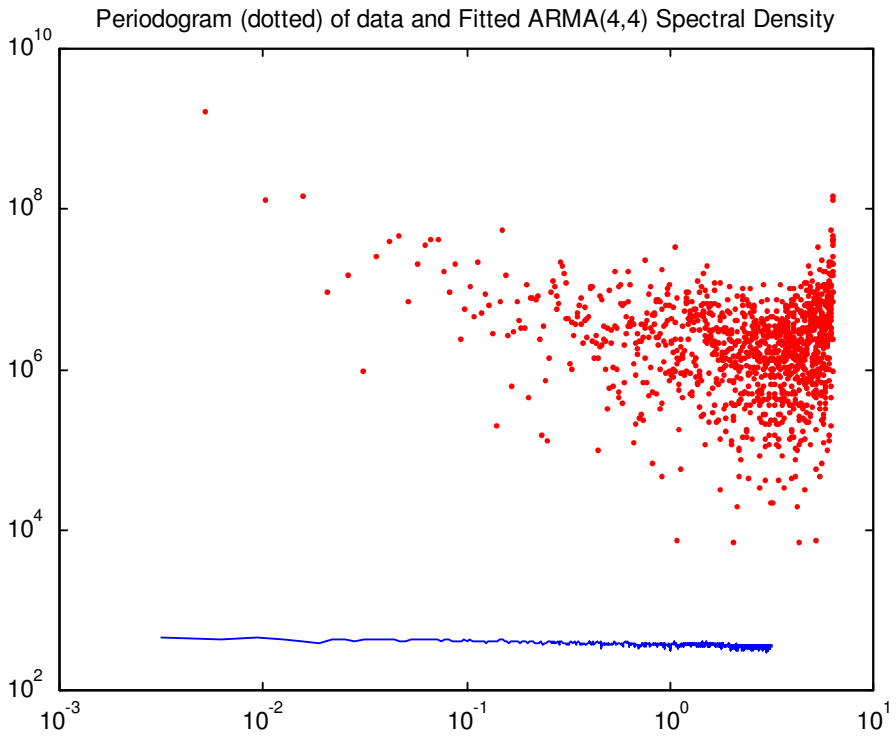
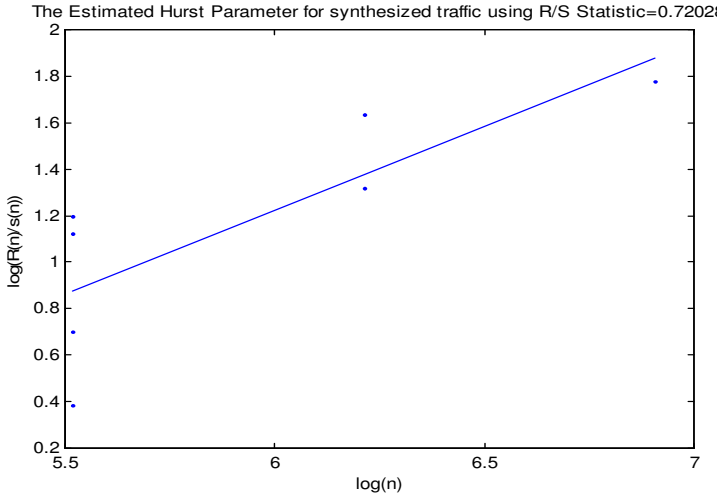


Fig 5.4. Periodogram and fitted ARMA (4,4) spectral density.

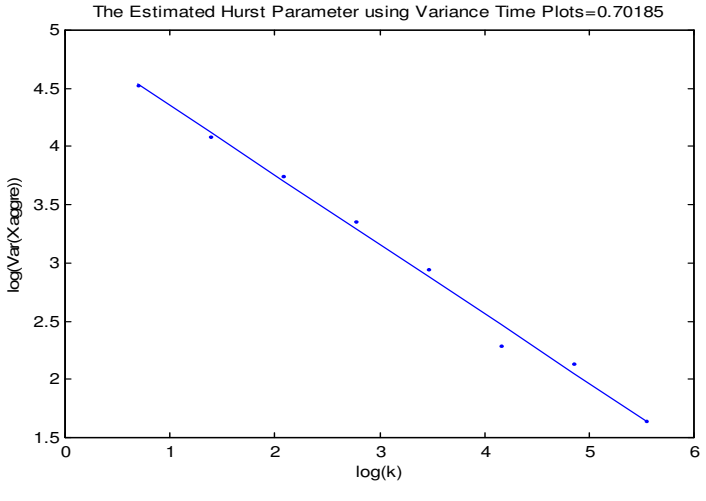
5.5.2 Synthesis of FARIMA (p,d,q) process

Using the Hosking's Algorithm discussed in section 5.4.3, a *MATLAB* code is written to generate a time series that is asymptotically self-similar *(See Appendix for the Code)*. The code generates a *FARIMA(0,d,0)* process with only Long-memory Properties. To incorporate Short-Range properties, the time series will be filtered through an *ARMA(p,q) filter.*

To see whether the *artificial* time series resembles the collected Internet packet traffic sequences, different tests were carried on. One test is to measure the *intensity of self-similarity* through the *Hurst Parameter.* Estimation of the Hurst parameter on the synthesized traffic yields results well in the range $(0.5 < H < 1)$. Figure 5.4 displays the result of the Hurst parameter estimation of R/S statistics and Variance Plots. The results are 0.72028 using R/S and 0.70185 using Variance plots. These results suggest that the synthesized traffic have the expected self-similar nature.



(a)



(b)

Fig. 5.4 Hurst parameter estimation of the synthesized traffic (a) R/S method and (b) Variance plot method

The Auto Correlation of the synthesized traces does have a structure that decays *slowly*, unlike the exponential decay, and similar to the collected Internet traffic traces. This is the property we call Long-Range Dependence. This property is depicted in Figure 5.5.

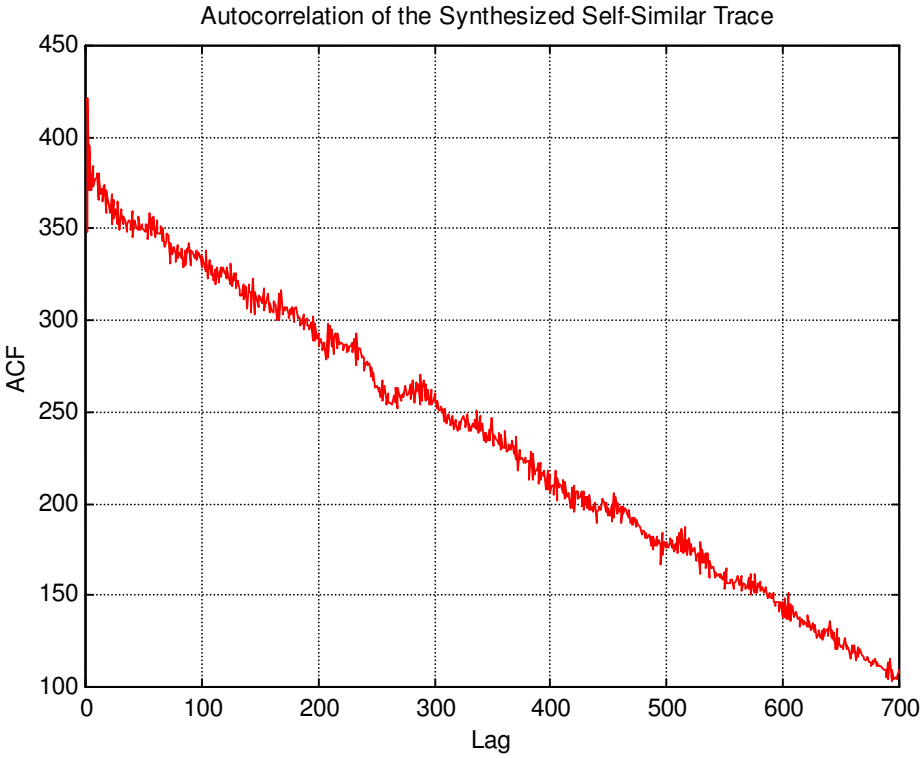


Fig 5.5 Autocorrelation of the synthesized FARIMA(0,0.4,0) process

Chapter 6

Self-Similarity and Internet Performance

6.1 Introduction

As asserted in Chapter 4, properties of the actual Internet traffic are very different from that predicted by traditional teletraffic models, such as Poisson processes. For the actual measured traffic, the correlation in traffic extends to a wide range of different time scales, or mathematically, the correlation function of the realistic traffic decays with lag in the way of a *power law*, which is the property of Long-Range Dependence (LRD), while for traditional model generated traffic, its correlation function decays exponentially fast, namely, Short-Range Dependence.

An obviously visible physical phenomenon observed in the measured Internet traffic is *Burst*, which appears in the counting processes of traffic as packet tend to come in clusters. Due to LRD, the burst phenomenon can not be smoothed out by simply aggregating the traffic in a larger time scale, which means even using very large time units to construct the counting processes, the burst phenomenon can still be observed. This demonstrates that the underlying process of measured traffic is *Self-Similar*. In *Chapter 4* a model to generate self-similar processes is developed.

The *performance* of the network is, therefore, dominated by *Self-Similarity.* As discussed in section 3.7.2, *Internet Packet-Round Trip Delay,* which is an important measure of *network performance,* seen as a *time series* also exhibited Long-Range Dependence. The delays experienced are *bursty* and hence Self-Similar processes should be employed to correctly *catch* their property.

From the traditional services of *electronic mail*, *file transfer*, and *remote login*, a lot of developments have been made on the Internet to support real-time, interactive multimedia services such as *packet telephony* and *video conferencing*. For these newer services, the delay characteristics of the Internet are of vital interest.

Quantifying delay is essential to build the next generation of multimedia services. Specifically *delay* and *delay variation (jitter), a*nd *packet loss* have a fundamental impact on the Quality of service (QoS) of a Real-time application. Understanding and modeling Internet delay and loss is a prerequisite to predicting the viability of real-time interactive services on the Internet.

In Internet a packet route generally includes a *tandem* of many Queues (which are as many as *hops*). As mentioned in *Chapter 1*, being fixed the size and same routing of a packet, the randomness (variation) the packet delay is by large due to the *Queueing performances* on the route, so the packet round-trip delay process is mainly determined by these *performances.* Understanding Queueing performance in relation to the *Self-Similar* nature of the traffic is considered crucial in quantifying delays.

TCP/IP, the most widely used protocol suite in the Internet, has TCP as a reliable transport layer protocol that provides connection-based transportation. With TCP there are several *time-out parameters* needed to be carefully tuned: the most sensitive one is *Retransmission Time Out (RTO)* parameter determined *dynamically by packet round trip delay in the Internet.* RTO is used to indicate when a packet considered lost in the Network by the sender and consequently invoke a *retransmission* event. If retransmissions occur continuously, the Network *utilization* will decrease remarkably due to TCP rate control algorithms.

In this chapter, *four* experimental *results* of this research work are presented. All researches are related to network performance, and the approach is to investigate network performance in relation to self-similarity. Investigation of the statistical properties of packet round trip delays, with an emphasis given to its relation with network load, and queueing performance analysis to investigate how Queues behave as the self-similarity of packet arrivals increase are discussed in Sections 6.2 and 6.3. These investigations indeed show that Queueing performance degrades as the self-similarity of packet arrival process increases. Section 6.4 presents the performance of VoIP in relation to packet delays. Section 6.5 presents TCP performance analysis to investigate how TCP retransmission time out parameter is affected as the self-similarity of the packet round trip delay process increases. The results of this analysis showed that TCP performance degraded as the self-similarity of the packet round trip increases.

6.2 The Packet Round Trip Delay Process

6.2.1 Measurement and Definition

The calculation of packet delay needs *four* time stamps, namely, T_1, T_2, T_3 *and* T_4. When a computer sends out a packet, it records the *leaving time* (T_1) on the out going packet. As the packet gets to the peer, the peer records the *arrival time* (T_2) on the packet; then the peer passes back the packet and records the *leaving time* (T_3) on the returned packet. The sender records down *receiving time* (T_4) when the packet arrives back.

So a Packet Round Trip Delay can be calculated as

$$T = (T_4 - T_1) - (T_3 - T_2) \qquad (6.1)$$

A packet round-trip delay is the *sum* of delays on each subnet link traversed by the packet. Each link (or hop) in turn consists of *four* components, including *Processing* delay, *Queueing* delay, *Transmission* delay and *Propagation* delay. So a Packet Round Trip Delay *process,* $T(t)$, is a *random variable* at time t. $T(t)$ describes the *packet round trip delay (Stochastic) process.*

Note: It is believed that a packet one-way delay process is similar to a round-trip delay process. In order to accurately measure packet one-way delays, one must synchronize the clocks of the packet source and destination to some degree of accuracy. [17]

6.2.2 Experiments

The *Ping* program (*refer to section 2.3.1.1)* is a candidate for catching these delays. The Ping program sends an *ICMP* (Internet Control Message Protocol) *echo request* message to a host, expecting an ICMP echo reply to be returned. Ping measures Round Trip Times (or RTT, used to represent Packet Round Trip Delays in TCP/IP's context) by storing the time at which it sends the echo request in the data portion of the ICMP message. When the reply is returned, it *subtracts* this value from the current time.

I used a custom application developed in Microsoft Visual Basic, named *VBPing* (© Mal Maliska, a *freeware* software downloaded from the Internet), that uses TCP/IP's, ***Ping*** application to calculate and record packet round trip delay. The recorded Packet Round Trip Delays or Round Trip Times (RTT) are used in the statistical analysis.

The goal of this part of the research work is to study the *nature* the Packet Round Trip Delay Process $T(t)$. This is achieved through studying the characteristics of a time series T_i. It is obtained by *discretizing* $T(t)$ with *t*. Simply, T_i is a *sample process* of $T(t)$ at $t = t_i$ where $i = 1,2,3,.....$ In our case t_i is the time at which the ping echo reply is received.

I have made different measurements of RTT at different times of the day for studying the statistical property of the Packet Round Trip delays. For the analysis, I considered only the replied Ping echo values as a time series disregarding *lost* Ping requests (*called IP Request Time Outs)* as they are not related to RTT. The *IP Request Time Out* Ping replays will be used in analyzing the performance of TCP (covered in *Section 6.4).*

The ping client is a Workstation in my office (*ICON Networks)* and the Ping servers are *yahoo.com (IP Address: 64.58.79.230)* and Ethiopian Telecommunication Corporation's Internet *DNS server* (*IP Address:* 196.27.22.43. The ping interval is *5 Seconds.* The Round Trip Times measured in millisecond get collected and arranged in a time series and their statistical behavior studied.

6.2.3 Statistical Analysis

As discussed in earlier chapters, traditional teletraffic models failed to catch the property of Internet Traffic. Therefore, my approach towards the statistical characterstics of Packet Round-Trip Delay process is to check whether they exhibit *Long-Range Dependence or not*. This Self-Similarity nature can be characterized by the value of the Hurst parameter, as discussed in *Chapter 3*.

6.2.3.1 LRD in Packet Round Trip Delay Processes

When the Packet Round Trip Delay (*or Round trip time or RTT)* aggregated in different levels, it maintains its *Bursty* nature. In other words, packet round trip delay is highly variable. This property is presented in Figures 6.1 and 6.2.

To investigate the *effect of Network Load* on the LRD of Packet Round Trip Delays, I ping ETC's DNS server (*IP Address:* 196.27.22.43) during high traffic time (10-12AM) and negligible traffic (3AM-4AM). This hour demarcation is based on the local population trend, since the server is located in Addis Ababa. The Hurst parameters estimated for these different time series show a significant variation, near to *1* during busy times or *High Network load* and near *0.5* during negligible traffic times.

6.2.3.2 Summary of Results

Figures 6.1 and 6.2 depict the *Scale-invariant* nature of the Packet Round Trip Delay process. The two figures display data pinged to the two ping servers. Each is re-iterated in *four* aggregation levels.

Figures 6.3 and 6.4 display the Hurst parameter estimation results using Variance-Time plot. As can be seen from results displayed in Figure 6.3 there is a variation in the intensity of self-similarity, as measured by the Hurst parameter, for the *busy* and *less busy* periods. *Figure 6.4* displays the estimate of the Hurst parameter estimation for the *yahoo.com* data using Variance-Time plots.

Figure 6.5 displays the Autocorrelation functions of the two Packet Round Trip Delay Processes. As can be seen, the Autocorrelation decays slowly (*as a Power Law)*, in contrast to exponential decay expected from traditional models.

From these observations, the following *conclusions* are derived:

- The *variability* in the Packet Round Trip Delays are observed scale-invariantly, they exist in all levels of aggregation; and not smoothed by having a larger time unit of observation and aggregation. The data sets look similar in the distribution sense. Hence, we can decide that *Packet Round Trip Delay process is statistically Self-Similar.* The autocorrelation function doesn't decay exponentially, rather a slow decay

is observed. Hence the data set of the Packet Round Trip Delay process has *Long Memory, or is Long Range Dependence* (LRD).

- The degree of *Self-Similarity*, as measured by the Hurst Parameter, of Packet Round Trip Delay process is dependent on the Network Load. As the utilization increases the Hurst parameter will tend to be close to 1.

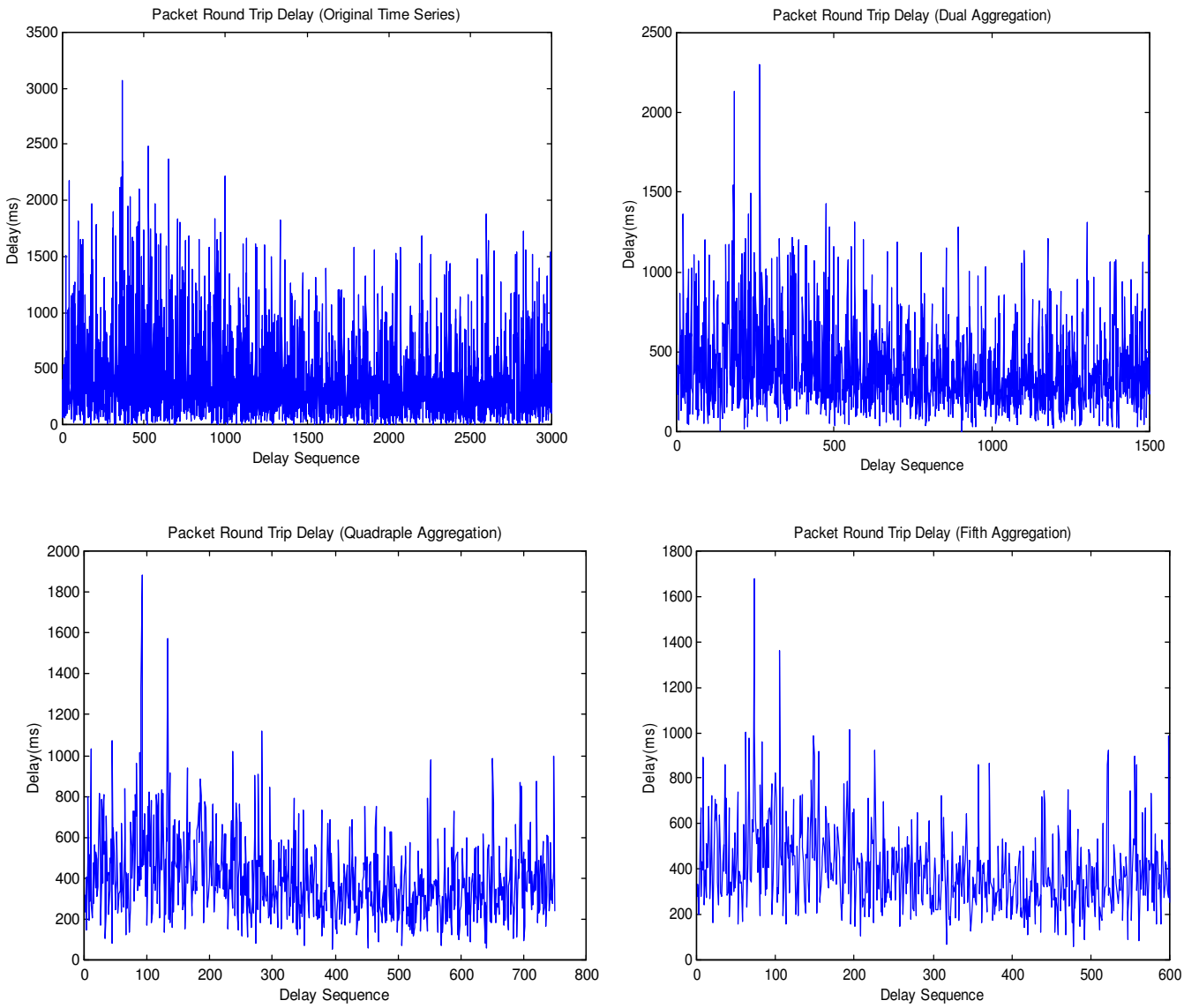


Fig. 6.1 Packet Round Trip Delay (ms) result of ping to ETC's DNS Server, displayed in four levels of aggregation

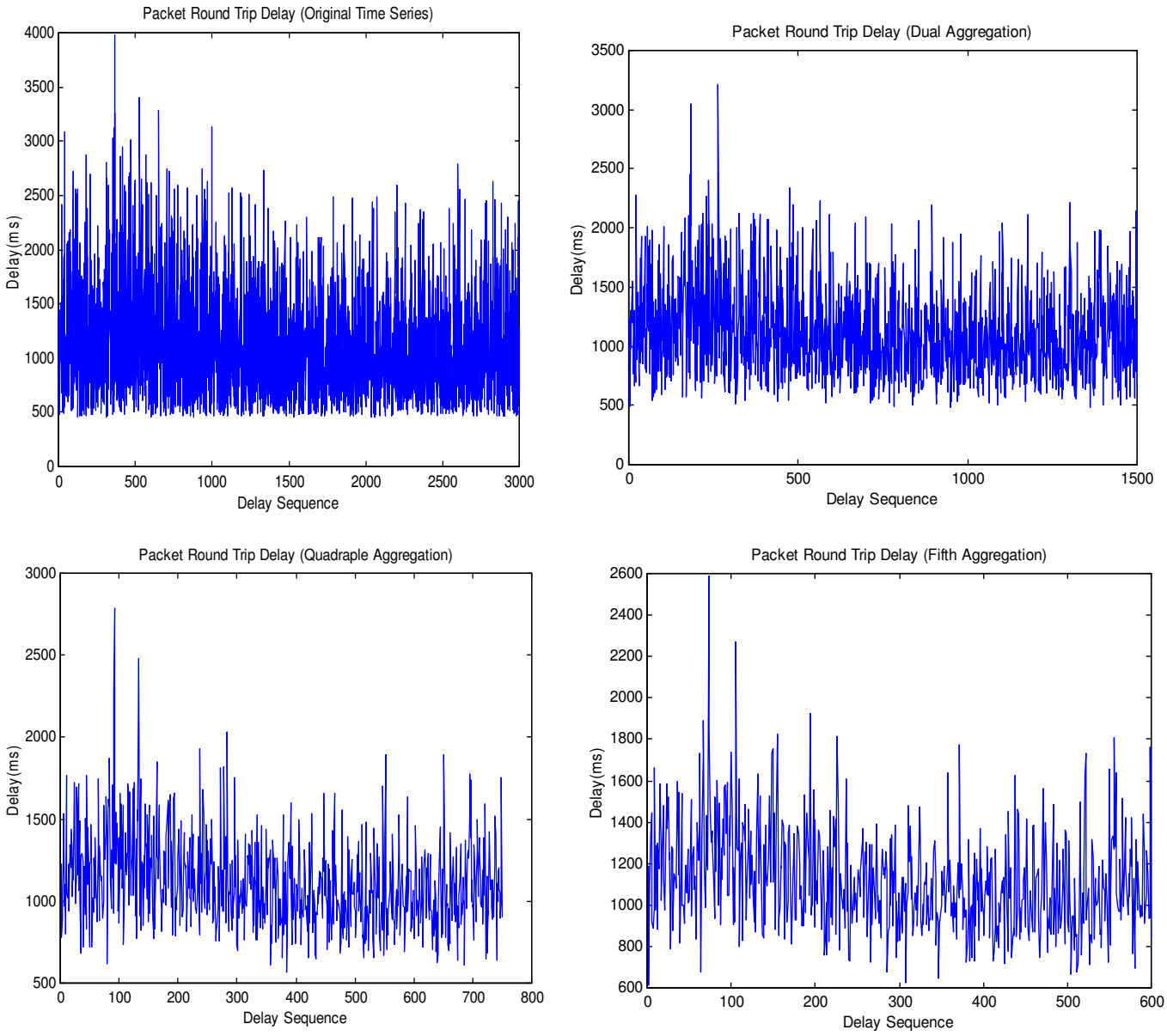
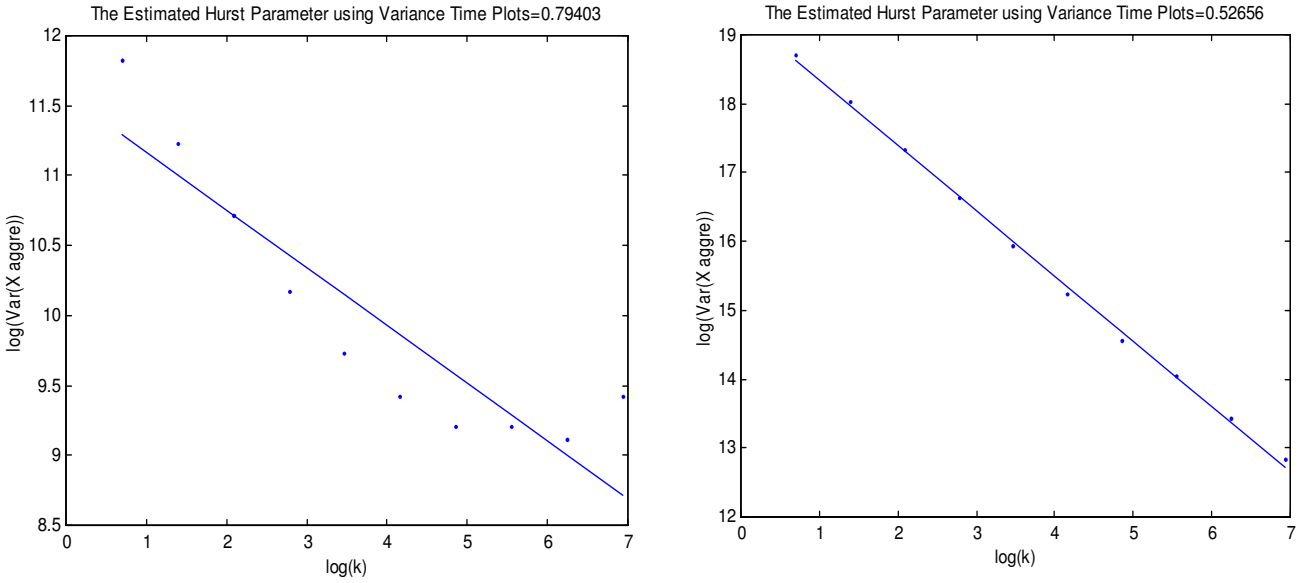


Fig. 6.2 Packet Round Trip Delay (ms) result of ping to yahoo.com, displayed in four levels of aggregation



(a) *(b)*

Fig. 6.3 Hurst Parameter estimates of the Packet Round Trip Delay of the ETC's DNS Server (a)During busy hours, H=0.79403 (b) During less busy hours, H=0.52656

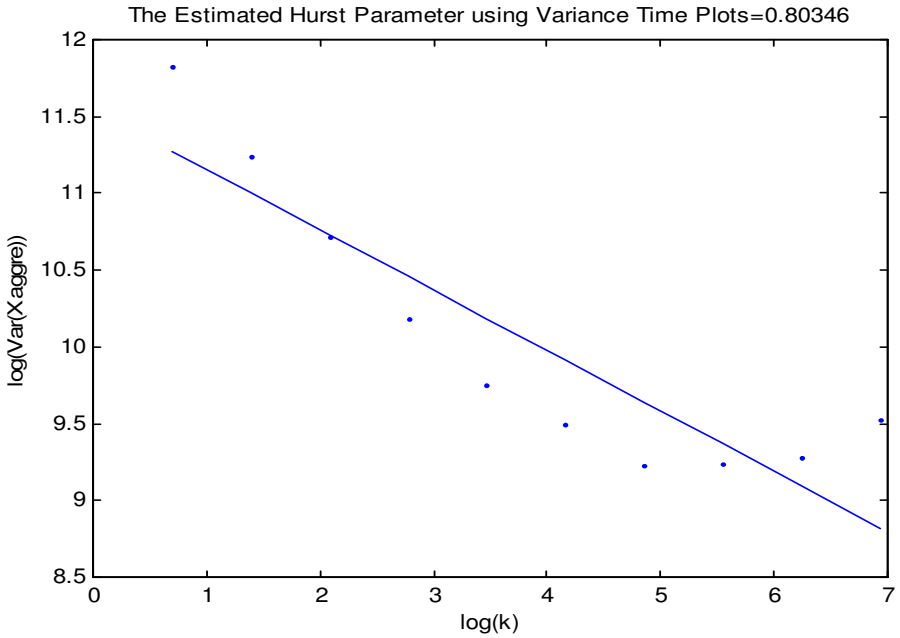
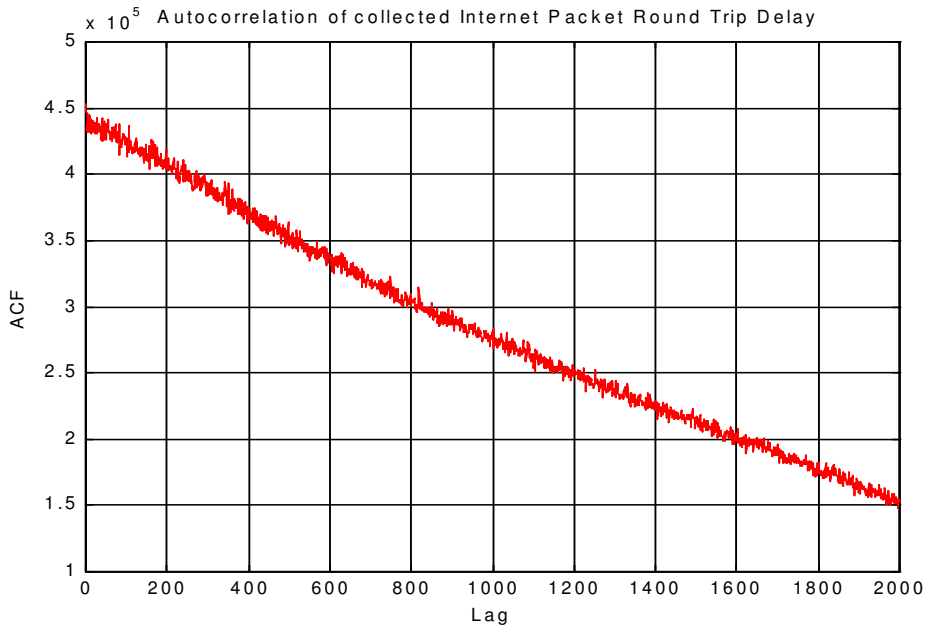
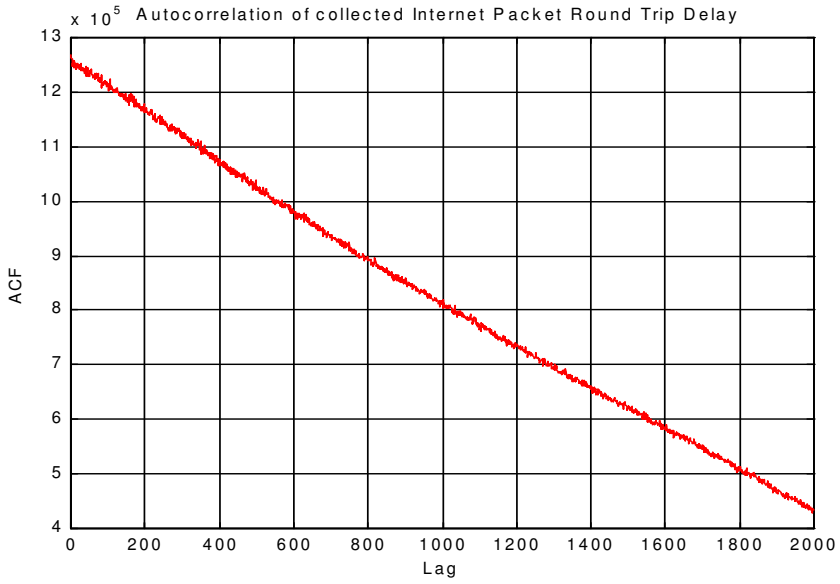


Fig. 6.4 Hurst Parameter estimates of the Packet Round Trip Delay of the yahoo.com Packet Round Trip Delay, H=0.80346



(a)



(b)

Fig. 6.5. Autocorrelation function for the first 2000 lags of (a) the ETC DNS and (b) the yahoo.com ping Packet Round Trip Delays.

6.2.4 Cause of Packet Round Trip Delay Self-Similarity

As noted in *Section 6.2.1*, a packet round trip delay in Internet consists of *four* components. Being fixed the size and route of a packet, the packet delay only changes with Queueing delays on the route. In Internet a packet route generally includes a *tandem* of many queues (as many as *hops*), so the packet round trip delay process is mainly determined by the Queueing performance along the route.

Expressing the packet round trip delay process $T(t)$ as the form.

$$T(t) = T_o + \Delta T(t) \qquad (6.2)$$

Where T_o is the *constant* part of a packet round trip delay. For a measured data, T_o is simply the *minimum* packet Round-Trip delay for a route. In this case, a packet with the minimum round-trip delay means, when the packet goes through all the links along its route does not meet any other packet ahead of it, all the Queues in the nodes along the route at the moment when the packet passes them are empty.

$\Delta T(t)$ is the *variable* part of a packet round-trip delay, which is the *sum of all the Queue delays* along its route.

$$\Delta T(t) = \sum_{i=1}^{N} Q_i(t) \qquad (6.3)$$

Where N is the total number of Queues along the packet routes, $Q_i(t)$ is the delay of the packet experienced in Queue *i*.

As discussed in *Chapter 4*, in an idealized network, cyclic On/Off transmission sources, which draw their holding times from a *heavy tailed distribution* (such as the *Pareto* distribution), will produce a self-similar process when multiplexed on to a Network link. If we consider a Network Router as a *Single Server Queue,* where the input rate is determined by the aggregate of a number of heavy-tailed packet-burst processes, then the Queue's length, when viewed as a time series, will be *self-similar.*

The degree of self-similarity measured for round-trip packet delay (estimated by *H*) will be based on the degree of self-similarity that exists at each router that the packet passes through. The degree of self-similarity at each Router is indicative of the burstiness of that Router's queue length over *different time scales*. When *H* is near 1.0, we would expect that the Queue is very large for long periods of time. This would suggest that the *packet drop rate* for a round trip Internet route should be, to some extent, correlated with the value of *H* for that route.

Therefore, one can argue that the *cause* of packet round trip self-similarity is the self-similarity of Queue Distribution. And the LRD in arrival packet *traffic* possibly causes the

LRD in Queue Distribution (*That is, the increase in Hurst parameter of Internet Traffic is related to the increase in Hurst Parameter of Packet Round Trip Delays).*

6.3 Queueing Performance Analysis

Moving beyond the statistical nature of Packet Traffic and Packet Round-Trip delays, in this section I tried to investigate the *impact of self-similarity on Queueing performance*. As mentioned in section 6.2, a packet delay in Internet consists of four components, out of which, Queueing delay is the most dominant(and the *only stochastic)* factor. In Internet a packet route generally includes a tandem of many queues, so the packet round-trip delay process is mainly determined by the Queueing performances along the route. Therefore, studying the *effect of Self-Similarity of arrival traffic on Queueing performance* is studying its influence on Packet Round Trip Delays.

While the research convincingly establishes the presence of LRD over a wide range of time scales in the packet traffic and delay processes, its implication to Queueing performance is a center of argument, in one hand arguments acknowledging LRD in packet traffic affecting Queueing performance, and in the other hand arguments emphasizing LRD has no practical impact and need not be incorporated into performance models. In this research it is tried to do experimentation, which demonstrates the first argument, that LRD in traffic indeed affects Queueing Performance.

6.3.1 Experimentation

This section discusses the experimentation performed to demonstrate the effect of Traffic Self-Similarity on Queueing performance. I used a Spreadsheet (*MS-Excel based)* Queueing simulator, *adapted* from a freeware Spreadsheet Queueing simulator engine, © A. Ingolfsson and T. A. Grossman. The simulator is *modified* to accept *Interarrival time* as an input series, instead of taking it from a fixed probability distribution. The service rate of the Queue is taken from a fixed probability distribution.

For the experimentation, two different data series were used. One is taken from the *Average Interarrival* time series of the actual collected Internet Traffic data. It is calculated from the collected traffic time series by dividing the time resolution (*10ms)* by the amount of packets, for the entire sequence. That is

$$Interarrival\ Time = \frac{10}{Number\ of\ packets}$$

Then it is normalized to fit the need of our simulator. Since the packet Interarrival processes is taken from the packet arrival process, it is expected to have long-memory. Indeed, the Hurst parameter estimated for the Interarrival time series has a value of *0.7231*.

The second data set is taken from a typical Short-Range dependent time series, generated from an ARMA (4,4) traffic generator, written in *Matlab* programming language. The estimated Hurst parameter for the series is 0.3.

6.3.2 Summary of Results

With *other conditions* kept the same, the simulator runs by *changing* the Interarrival time series from the two data sets mentioned in the previous section.

The resulting *average Queueing delay* for the *Self-Similar* Interarrival time sequence is significantly *larger (Value=0.10894)* than the resulting average Queueing delay of the *Short-Range* dependent Interarrival time *(Value=0.083).* Therefore, it can be concluded that Queueing performance is affected due to the *Self-Similar or Long Memory* nature of the packet arrival process, that is, *Queueing performance degraded as the self-similarity of the Packet arrival process increases.*

However, the experimentation discussed has some limitations such as the *amount of traces* implemented in the Interarrival time series and the failure of implementing *real-time* packet arrival on the simulator. A more exhaustive experimentation (resolving both the limitations of *Trace amounts and real-time nature)* can be done by using Real Network environment or *real-time Network Simulators* such as the LBNL-NS (Laurence Berkeley National Laboratory-Network Simulator-*the most commonly used Network simulator*)[3] as a simulation environment. *Kihong Park, et al [10]* studied the effect of traffic self-similarity on different aspects of Network performance, such as *throughput, packet loss, and Queueing delay*, using LBNL-NS. They used Heavy-tailed *file distribution* to generate aggregated self-similar network traffic (*refer section 4.5)* on the bottleneck link of which the Queueing delay to be computed. Their simulation results were obtained from several hundred runs of NS, each run executed for 10,000 simulated seconds, logging traffic at 10ms granularity. They used the tail index or shape parameter α of the heavy-tailed distribution (in their case Pareto Distribution) to control the self-similarity. From section 4.5, recall that $1 < \alpha < 2$ and $H = \frac{3-\alpha}{2}$. That is, when α is near 1 (i.e. H goes near 1) the traffic gets more self-similar. They did run tests for different *link buffer sizes* and set the result shown in Figure 6.7. As can be seen from Figure 6.7, the *mean Queueing delay* increases as the traffic gets more self-similar. This is in harmony with the experimentation I made.

[3] It is difficult to get a working version of NS on Windows platforms. It also requires mastering of the OTcl scripting language as well as a need for big computing power.

Note also from the figure, the *super-linear relationships* between link buffer capacity and Queueing delay. This has led to proposals advocating a resource provisioning strategy due to link buffer's simplistic, yet curtailing influence on Queueing: *if buffer capacity is small, the ability to queue or remember is accordingly diminished.* This resulted in a different QoS strategy for LRD traffic in such a way that keeping buffer capacity small and increasing bandwidth to boost Queueing performance.

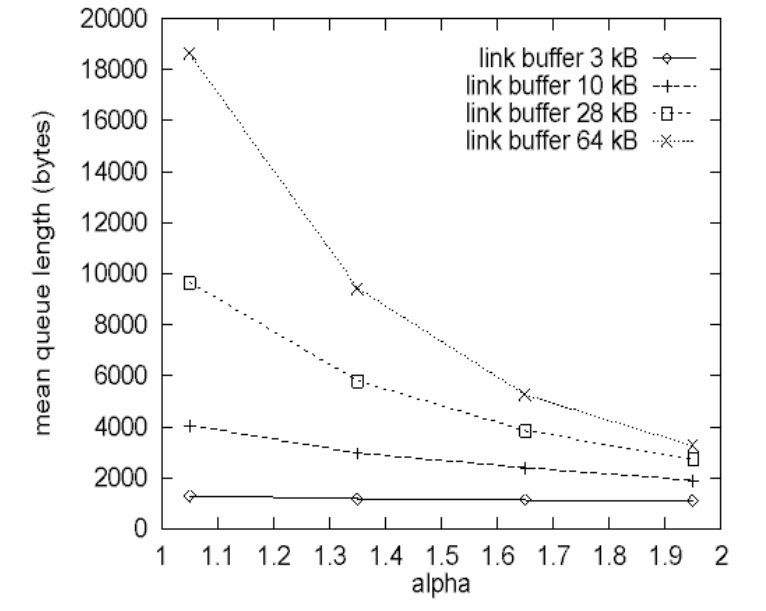


Fig. 6.7 Mean Queue length as a function of α *(From Reference [10])*

6.4 VoIP Performance Analysis

The Internet will become a ubiquitous infrastructure, used by numerous applications having various requirements and that generate traffic that has different characteristics.

Voice requirements are stringent, toll-quality real-time communication is needed, which limits the maximal tolerable round-trip delay to 200-300 ms; that is, one-way delay must be in the range of 100-150 ms for adequate performance. *Delay* and *jitter* are the important

measures of QoS particularly for the voice traffic in the network environment with bursty background traffic.

6.4.1 Delay

Internet traffic is proved to have bursty nature. That was explained through the *high variability* and *correlations spanning large time scales,* which is tackled through *Self-similarity and Long-range dependence.* The delay process associated with the traffic is also proved to have the *Self-similar* nature. The *high variability* encountered both in *traffic and delay processes* affects the performance of the applications carried over the Internet. Also we have proved that Queueing performance degrades as the self-similarity of the packet arrival process increases. This implies that packet traveling with a self-similar traffic will encounter more delay than the case when the traffic is non self-similar or short-memory. The delay encountered as a packet carried on the Internet has more impact on real-time traffic, such as VoIP. To achieve a reliable real-time communication, issues related to the very nature of the Internet have to be resolved. That way the *migration* of PSTN to VoIP can be achieved.

For a given voice connection, the only *random* component of voice delay is the *Queueing* delay in the network. As proved in section 6.3 the increase in self-similarity of the network traffic arrival implies Queueing performance degradation. VoIP carries voice over the Internet, whose traffic and delay processes are self-similar. Therefore, the queueing degradation is implied for VoIP packets, if they are treated in *unison* with the other Internet packets. Also discussed in section 6.3 was that average queueing delay for a given *buffer utilization* is inversely proportional to the available bandwidth, that is, delay percentiles *decrease* with an increase in available bandwidth, that resulted in a different QoS strategy for LRD traffic in such a way that keeping buffer capacity small and increasing bandwidth to boost Queueing performance.

Interms of traffic characterstics, voice streams have *low data rates* (in the order of tens of killo bits) and exhibit *low burstiness.* The approach of QoS strategy for LRD traffic, advocating the use of *large bandwidth,* is therefore not suitable for managing *voice traffic "specific"* QoS mechanisms, but suitable for the performance of the whole Internet. Because of its stringent requirement of delay and particular characterstics, voice traffic should be treated *differently* than other traffic in the Internet. This can be achieved by having voice traffic through a *separate queue*. If other traffic is flowing on the link, then voice traffic could be serviced using Priority Queueing (PQ), and given the highest priority over all other traffic. At the Internet scale, high priority can be provided to voice traffic in the *Differentiated Services* framework.

6.4.2 Jitter

Variable packet delays or jitters in a single voice stream will result in voice degradation. These delay variations are results of the Queueing delays encountered with the different packets that made up the voice stream. Therefore, if the *variations* in one-way delay are properly resolved, then the jitter problem will be resolved. Other wise, QoS mechanisms

should be implemented to counteract the delay variations. The usual implementation to resolve voice degradation due to jitter at the receiver side is to use *jitter buffer*. This is achieved by *holding* arriving packets until a later playout time in order to ensure that there are enough packets buffered to be played out *continuously*.

Following is an experimentation on how *packets (can be data, voice or video)* encounter *variable* delays as they traverse the routes of the Internet, assuming without any QoS measures. The delay variation or *burstiness* exists even if the packets follow the same route through out the stream duration, due to the random nature of Queueing delay encountered along the path.

Section 6.2 asserted the *high variability or bursty* nature of packet round trip delays. One-way delay, which determines the nature of jitter, is also believed to have such nature. To have an insight on the nature of the one-way delay, I use the Traceroute application (*refer section 2.3.2).* Traceroute lets us see the *route* that IP datagrams follow from one host to another and estimate the time spent while the packets traverse each hop along the path. There is no need of time synchronization, since the sender machine manages both timings. Although there are no guarantees that two consecutive IP datagrams from the same source to the same destination follow the same route, most of the time they do. Logging *one-way delay values* for some time and treat the trace as a time series shows how bursty the delay variation is. Figure 6.8 depicts this nature, which is obtained by tracing a path to ETC's DNS server (*IP Address: 196.27.22.43)*. Even if a packet traverse same path it will get delayed randomly, therefore, unless voice traffic enjoy *exceptional* handling, it will suffer voice degradation due to jitter. Recommended QoS mechanisms are discussed in section 3.4.2.

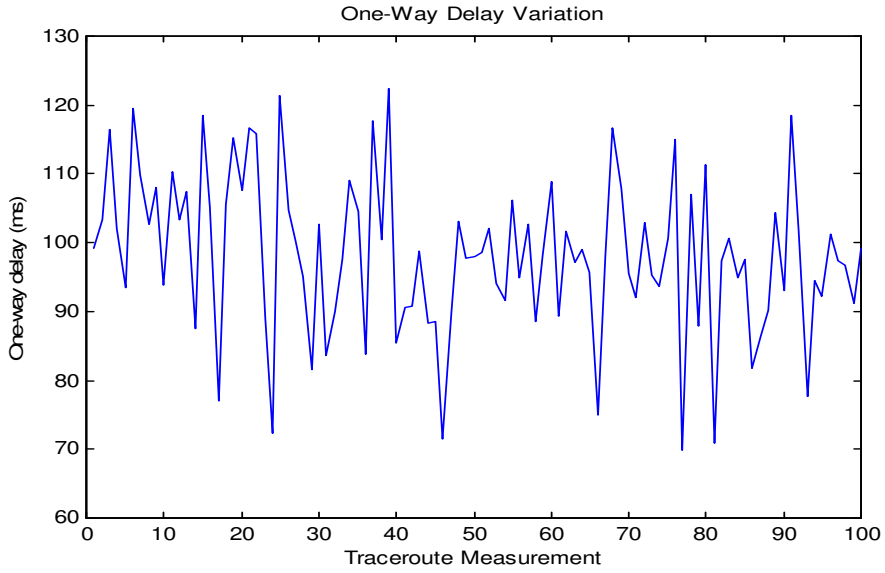


Fig. 6.8 One-way delay variations

6.4.3 Summary

The inherent self-similar nature of the Internet made the possibility of carrying *voice traffic* to be accompanied by voice quality degradation, of course, *unless* proper QoS measures are taken. Delays relating to the Queueing performance degradation in self-similar packet arrivals may get intolerable at the receiving side. And also the self-similar or variable nature of the packet delay process implies voice quality degradation due to inter-packet arrival variations.

For paving the way for VoIP gradually replace the existing PSTN, these issues should be resolved. The conclusion, which can be drawn from the discussions in the previous two sub-sections, is that; to achieve good quality of voice over the Internet, voice traffic should be handled *independently* of the remaining Internet traffic, through implementations of new features in the *existing* Internet infrastructure.

6.5 TCP Performance Analysis

This section tries to investigate the effect of packet round trip delay self-similarity on TCP's *throughput* performance, which is captured through the Retransmission Time Out (RTO) parameter. It is important to evaluate the impact because TCP is the most dominant *transport* protocol in the current Internet, and its performance depends on the packet round trip delay or RTT (Round trip time). Internet packet delay is Long Range dependent as proven in section 6.2. It is bursty across multiple time scales, which implies that end-user Quality of Service in the Internet is likely to be affected by long period of very large and/or highly variable delays.

6.5.1 TCP Transport Mechanism

TCP employs an *ACK* (*acknowledgment*)-based window control, that is, a *TCP sender* updates its congestion window size every time it measures RTT [15]. This measurement is taken *randomly.* Therefore, TCP *throughput performance* depends on the RTT. With the most basic mechanism in the Internet to detect losses, the sender re-transmit a segment if it's ACK has not been received in the expected amount of time (this is *RTO based on the RTT*). If segment losses are detected by RTO, TCP congestions window, which specifies the amount of data the TCP sender can transmit before receiving an ACK is reduced to one segment, which implies RTO based method effectively *reduce throughput*. Therefore, the goal of a good RTO estimator should be to *minimize* the number of unnecessary re-transmissions due to

failure in proper estimation of RTO. If RTT drastically changes, RTO may not be able to *adapt* to the RTT and the TCP sender may end up *retransmitting* segments.

In *Jacobson's algorithm* [16], RTO is calculated by

$$RTO = SRTT + kRTTVAR \qquad (6.4)$$

$$RTTVAR = RTTVAR + h(|SRTT - RTT_{meas}| - D) \qquad (6.5)$$

with $k = 4, h = 0.5$ in many cases, where $SRTT$ is a *smoothed* estimate of RTT and $RTTVAR$ is a smoothed estimate of the variance of RTT. Both variables are updated every time an RTT measurement, RTT_{meas}, is taken. $SRTT$ is updated using an exponentially weighted moving average (EWMA) with a gain of α.

$$SRTT = (1-\alpha)SRTT + \alpha RTT_{meas} \qquad (6.6)$$

The next section tries to investigate TCP performance as captured by *unreturned ACKs,* which are responsible for the amount of *bandwidth wastage* due to *retransmission traffic.*

6.5.2 Experimentation

As discussed in section 6.5.1, TCP employs a *Retransmission Time Out (RTO)* parameter to re-send a packet for which it didn't get an *ACK* (Acknowledgment). TCP dynamically re-calculates the value of the *RTO* based on the measured *Round Trip Time (RTT),* which is equivalent to the *Packet Round Trip Delays.* Due to *the high variability or bursty nature* of RTT, TCP may invoke *premature retransmission time out* and *waste useful bandwidth on the link.*

In this section, a study that catches the percentage of *RTO events*-invoked whenever the sender failed to receive ACK packets will be discussed. The relation between the occurrence of RTO and *Packet Round Trip Self-Similarity* is also to be analyzed.

For studying the relation between occurrence of RTO and self-similarity of packet round trip delays, I used a *simple* experiment using the *Ping* application, which was also used in section 6.2. *Ping* uses ICMP echo request message to a host expecting an ICMP echo reply to be returned. If the echo doesn't returned in the pre-specified *duration (depends on the implementations of the Ping program),* the application issues an *"IP Request Time Out"* message. This may happen in two cases: the first case is when a packet get *lost* due to congestion on the Queues along the route, and the second is if a packet encountered a *prolonged delay* at the intermediate Queues. Unless mechanisms are implemented within the Ping application to track the variability *or burstiness* of the Packet Round Trip Times and dynamically update the pre-specified duration, the amount of premature "IP Request Time Out" messages get increased along with an increase in traffic.

In a similar fashion, *TCP* requires ACK packets to arrive from the receiver to determine whether a packet gets delivered. Otherwise it will keep re-sending the same packet. TCP employs RTO to invoke a packet retransmission event, but implements ways for dynamically calculating *RTO* from randomly measured RTT values. The failure in getting ACK message may happen either due to *packet loss* or *prolonged delay*. These cases are similar to the events we catch using the *Ping* Application. Therefore, having knowledge of the *percentage* of the *"IP Request Time Out"* messages gives a *rough* estimate on how TCP performs on that link.

I perform a continuous *Ping* operation using the custom application mentioned in section 6.2 to the *yahoo.com server (IP Address: 64.58.79.230).* Then the percentage of the *"IP Request Time Out"* messages out of the replies to the total *Ping* requests is calculated. By *taking out* the "*IP Request Time Out"* events from the collected traces, I rearrange the successful *Ping* reply *Round Trip Times* as a time series, then the corresponding *Hurst parameter* of *this* time series estimated.

6.5.3 Summary of Results

Figure 6.9 shows the relation between the *Percentage of "IP Request Time Out" messages and the Hurst Parameter of the Packet Round Trip Value.* From this observation, it can be concluded that, a*s the self-similarity of the Packet Round Trip Delays increases, the wastage in bandwidth due to packet re-transmissions is increased.*

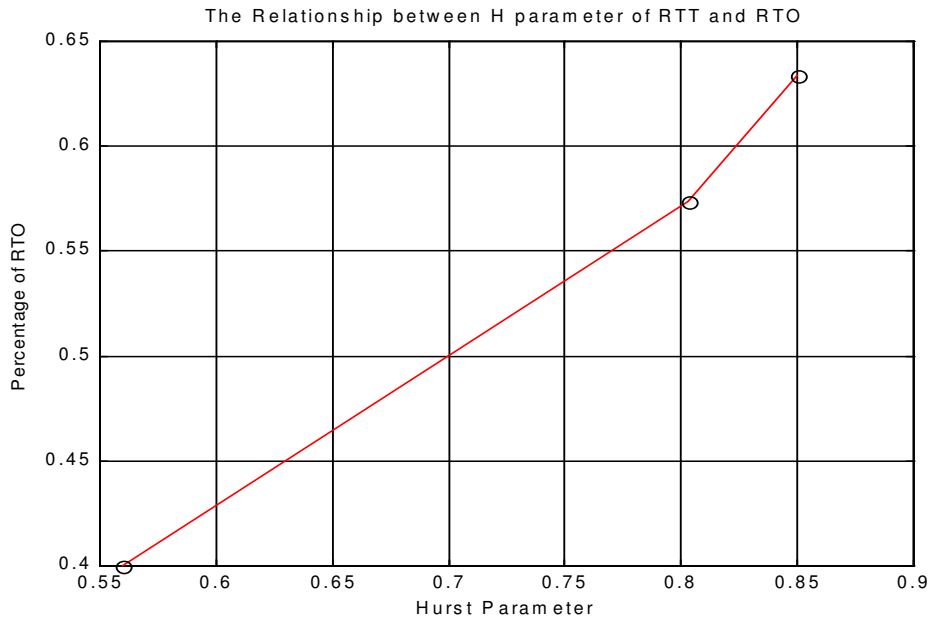


Fig. 6.9. Hurst Parameter versus Percentage of Retransmission

The experimentation is not exhaustive in the most part due to the *indirect* measure of TCP's performance through another application of the TCP/IP protocol suite and failure of simulating a *real-time* environment. In a similar manner to Queueing performance analysis, the limitations can be improved by using Real Network environment or real-time Network Simulators. Tasuyara Hagiwara, et al [16] did research on the effect of Packet Round Trip Delay or Round Trip Time (RTT) self-similarity on the performance of TCP using the LBNL-NS simulator to simulate a network environment, with all the links using TCP connections. The TCP *senders* send files according to the Pareto Distribution (*refer back section 4.5)* to create aggregated self-similar traffic, and RTO estimations are evaluated using different Hurst parameters of RTT to investigate *premature time outs.* RTO at *H* values near *1* is found to be *burstier* than RTO at *H* values near *0.5*. Figure 6.10 shows this finding. *This result agrees with the conclusion I made.*

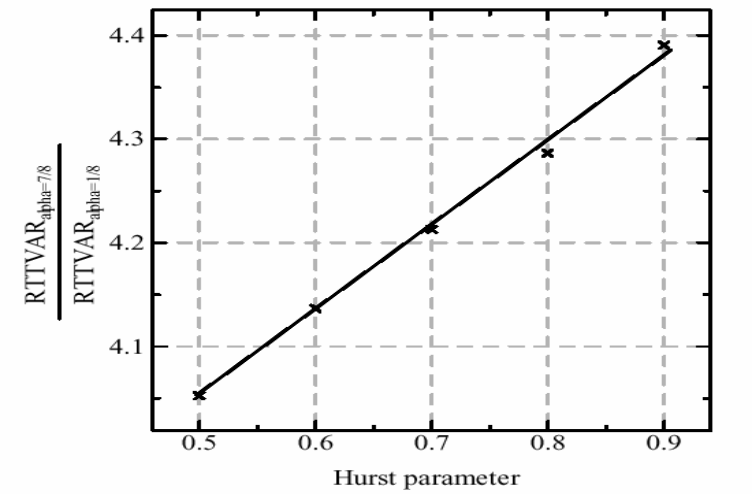


Fig. 6.10 Hurst parameter effect on retransmission of TCP *(From reference [11])*

Chapter 7

Conclusions and Recommendations

In this research, Study of Internet Traffic and Packet Round Trip Delays, Model Selection and artificial synthesis of self-similar processes and Study on the effect of Self-Similarity on Network Performance are covered. With proper statistical and mathematical tools and computer simulations, the following main findings are taken out as results.

- *Internet traffic* and *Packet Round Trip Delay processes,* as seen as a time series, are *statistically Self-Similar.* The autocorrelation function doesn't decay exponentially, a slow decay is observed. Hence the data sets have *Long Memory, or are Long Range Dependent* (LRD).
- The degree of *Self-Similarity*, as measured by the Hurst Parameter, of Packet Round Trip delay process is dependent on the Network Load. As the utilization increases the Hurst parameter will tend to be close to unity.
- *FARIMA* process is proved to model self-similar processes. It is capable of *simultaneous* modeling of Long-Range and Short-Range behavior of Network Traffic. A program is developed to synthesize self-similar traces based on Hosking's Algorithm.
- Queueing performance has been affected due to the *Self-Similar or Long Memory* nature of the packet arrival process, that is, Queueing performance *degraded* as the self-similarity of the Packet arrival process increases.
- As the self-similarity of the Packet Round Trip Delays increases, *TCP throughput performance* is degraded. As the Hurst parameter of the Round Trip Time becomes near to unity, the wastage in bandwidth due to packet re-transmissions is increased.
- The self-similarity of Internet degrades voice quality in VoIP. Both *delay and delay-variation or jitter* behaves erratically due to the self-similar nature of packet arrivals and delays. To achieve good quality of voice over the Internet, voice traffic should be handled *independently* of the remaining Internet traffic, through implementations of new features in the *existing* Internet infrastructure.

Important implication of the long-range dependence and scale invariant "burstiness" of the Internet traffic is drastically different from both conventional telephone traffic and from stochastic models for packet traffic usually considered in the literature (Example is the *Poisson* Model). Traffic models for voice traffic, developed over the years to serve the telephone network, did not apply as might have been hoped because voice traffic does not give rise to the same traffic characteristics as Internet data traffic, which is *burstier*. Fractal models, such as the FARIMA model proposed in this research, should be used.

With the help of the findings and developed model of this research work, one can extend this work in the following areas:

- *Physical modelling of self-similarity:* Developing physical models that can explicate traffic characterstics in terms of elementary, verifiable system properties and network mechanics. This will give a clear insight to the mathematical and statistical findings asserted in this work.
- *Traffic control for self-similar traffic*: It can be exploited on two fronts: One as extension of performance analysis in the resource provisioning context and the other from the multiple time scale traffic control perspectives where correlation structure at large time scales actively exploited to improve network performance. The developed model in this work can be used in this regard.
- *Congestion control for self-similar network traffic*: This can be facilitated by using the long-term correlation structure present in long-range traffic for congestion control purposes. The long-range autocorrelations can be exploited in further research using the artificial traffic synthesis program developed in this work.
- *Wavelet analysis of self-similar or scaling phenomenon*: Due to their ability to localize a given signal or time series both in time and scale (or frequency), wavelets provide a powerful and refined technique for detecting and quantifying scaling behavior in measured traffic.
- *Second order performance measures under self-similar conditions*, such as impact of self-similarity on jitter and impact of self-similarity on packet level Forward Error Correction mechanisms. This will have an impact for multimedia data transport and QoS

provisioning. The Queueing performance analysis done in this work can be used as a starting point.

References

1. W. E. Leland, M. S. Taqqu, and D. V. Wilson, "*On the self similarity nature of Ethernet Traffic"(Extended Version*)", IEEE/ACM transactions on Networking, 1994.
2. B. B. Mandelbrot, "*The Fractal Geometry of Nature*", Freeman, New York, 1983
3. "*Self Similar Network Traffic and Performance Evaluation",* Edited by K. Park and W. Willinger, John Wiley and Sons, 2000
4. A. Adas and A. Mukherjee, "*On Resource Management and QoS Guarantees for LRD Traffic",* IEEE InfoCom'95, PP779-787, 1995
5. R. Addie, M. Zukerman, and T. Neame, "*Fractal Traffic, Measurement,Modelling and Performance Estimation",* IEEE InfoCom'95, PP 977-984, 1995
6. Walter Willinger and Vern Paxon, "*Where Mathematics Meets the Internet",* IEEE/ACM Transactions on Networking, Vol. 3, No. 3 1995.
7. J. Beran, "*Statistics for Long Memory Processes"*, Monographs on Statistics and Applied Probability, Chapman and Hall, New York, 1994
8. D. R. Cox, "*Long Range Dependence: A Review",* Iowa State University Press, Ames, 1984
9. Krishnamurty Nagarajan, "*Fractional ARIMA Process and its Implication in Network Traffic Modelling",* Ph. D. Thesis, Georgia Institute of Technology, 1998
10. K. Park, G. Kim and M. Crovella, "*On the Effect of Traffic Self-Similarity in Network Performance", 1997 SPIE,* International Conference on Performance and Control of Network System, Nov 1997.
11. J. Davidson and J. Peters, "*Voice Over IP Fundamentals",* Cisco Press, Indianapolis, 2000.
12. Monson H. Hayes, *"Statistical Digital Signal Processing and Modelling",* John Wiley & Sons, Inc., 1996.
13. M. Garret, W. Willinger, *"Analysis, Modelling and Generation of Self-Similar VBR Video Traffic",* SIGCOMM, Networking Transaction, 1994
14. M. Borrela, S. Uludag and G. Brewster, *"Self-Similarity of Internet Packet Delay",* IEEE ICN'97, pp 513-517, Jan, 1997
15. W. R. Stevens, *"TCP/IP Illustrated, Volume 1",* Addison Wesley, New York, 2000
16. T. Hagiwara, H. Mashima, T. Matsuda, and M. Yamato, *"Impact of Round Trip Delay Self-Similarity on TCP performance",* Osaka University, 1999
17. M. S. Borrela, G. B. Brewster, *"Measurement and Analysis of Long-Range Dependent Behavior of Internet Packet Delay", IEEE/INFOCOM, pp497-504, 1997*
18. Cisco Press, *"Cisco AVVID QoS Design Guide",* Cisco Systems, San Jose, 1999
19. Z. Wang, *"Internet QoS: Architectures and Mechanisms for Quality of Service",* Book Review, An International Electronic Publication of the Internet Society, www.isoc.org/oti, July, 2002
20. W. Willinger, M. Taqqu, R. Sherman and D. Wilson, *"Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at Source Level",* ACM SIGCOMM, 1995
21. Athanasios Papoulis, *"Probability, Random Variables and Stochastic Processes",* McGraw Hill Book Company, International Edition, 1984