PROVISIONING INTERNET BACKBONE NETWORKS TO SUPPORT LATENCY SENSITIVE APPLICATIONS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Charles J. Fraleigh

June 2002

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Fouad Tobagi
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Nick McKeown

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

David Donoho
(Department of Statistics)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Christophe Diot
(Sprint Advanced Technology Labs)

Approved for the University Committee on Graduate Studies:

To the memory of my mother

Gail Fraleigh

and the memory of my father

Peter Fraleigh

Abstract

Interactive applications such as voice, audio, and video, as well as business applications such as Virtual Private Networks are becoming an increasingly important component of Internet traffic. Such applications have strict requirements on the total end-to-end delay which may be incurred in the network. One approach to meeting these delay requirements, known as service differentiation, is to give preferential treatment to latency sensitive traffic. Doing so may lead to an efficient network design with the minimum amount of resources (e.g., bandwidth) required to support the needs of each traffic type. An alternative approach is to provide sufficient resources so that all traffic meets the most stringent delay requirements. This latter approach is known as overprovisioning.

In the context of wide-area Internet backbone networks, two factors make overprovisioning an attractive approach. First, the high link speeds and large volumes of traffic make service differentiation complex and potentially costly to deploy. Second, given the degree of aggregation and resulting traffic characteristics, the amount of overprovisioning required may not be very large. We establish that this is indeed the case by collecting and analyzing traffic measurements from the Sprint IP network, a

commercial Tier-1 Internet backbone.

We begin by performing network simulations using a set of 331 one-hour traffic measurements from the Sprint network. These simulations demonstrate that link utilization can reach 80% - 90% before queuing delays begin to exceed several milliseconds. While the simulations can be used to evaluate the level of overprovisioning required in the network, many network design problems are greatly aided by an analytic traffic model. We therefore develop a traffic model which captures the observed characteristics of backbone traffic and derive expressions for the delay through a single queue in the network, as well as the end-to-end queuing delay. Using this model we solve several network design problems including capacity planning and optimal route selection.

Acknowledgments

This thesis would not have been possible without the help, support, and guidance of many people. First, I would like to thank my adviser, Fouad Tobagi. I have learned a great deal from Fouad, especially in how to critically and methodically approach research problems. I would also like to express my gratitude to Christophe Diot, manager of the IP group at the Sprint Advanced Technology Labs. It has been Christophe's support and vision that has made the IPMON measurement system possible. I am also extremely grateful to my associate adviser, Nick McKeown, and to the other members of my orals committee, David Donoho and Antony Fraser-Smith.

It has been my privilege to work with two extremely talented groups of researchers over the past several years. The Multimedia Networks Group at Stanford, especially Wael Noureddine, Athina Markopolou, Benjamin Chen, Cristina Hristea, Mansour Karam, and Jose-Miguel Pulido, have provided invaluable feedback on my papers and this thesis. The IP group at Sprint, especially Sue Moon, Dina Papagianaki, Gianluca Iannaccone, Nina Taft, Alberto Medina, and Supratik Bhattacharyya, have been a pleasure to work with and have helped immensely in the development of the IPMON project. I would also like to especially thank Ed Kress, Imed Chihi, and Rich

Gass for taking over responsibility for the day-to-day operation and deployment of the IPMON systems. I don't think I would have made it through the Ph.D. process if they had not been available to help.

There are several other people who I would like to thank for their support and friendship throughout my time at Stanford. Nabeel Ibrahim, Bob Schaffer, Stephanie Schaff, and Michele and Andy Momotiuk have made these past years some of the most enjoyable of my life. I also would like to thank some of my good friends from my days at Purdue and before whom I always will remember: Dave and Julie Albert, Mike Mshar, Steve Doughten, Tammy Keith, and Mike and Julia Seiler.

Finally, I would like to thank my wife Lori. Her love and support has been the foundation of this work and for all of my other accomplishements. I am indeed fortunate to have found such a wonderful person with whom to share my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Internet History and Architecture

The origins of the Internet can be traced back to a network known as the ARPANET which was developed to allow university researchers to access large mainframe computers. In the late 1960s, mainframe computers were being requested by many researchers supported by the U.S. Department of Defense Advanced Research Projects Agency (DARPA). These computers represented specialized hardware and software resources. For example, one computer system would support symbolic mathematical analysis programs while another computer would support network simulation and modeling. To allow many researchers to share these specialized resources, DARPA initiated development of a data network known as the ARPANET [66]. In 1969 the first four nodes of this network were installed at UCLA, the Stanford Research Institute, the University of California Santa Barbara, and the University of Utah. Over the next 30 years, this network evolved into what is known today as the Internet.

At the time at which the ARPANET was developed, communication networks used a technique known as *circuit switching*. In a circuit switched network, users establish a dedicated connection with a fixed amount of bandwidth between the source and destination for the duration of their communication. This approach is efficient for traffic such as telephone voice calls which transmit data at a constant bit rate and whose connections durations are longer the amount of time required to establish the connections.

Computer data traffic, however, is traffic is typically very bursty. The rate at which traffic is generated fluctuates over time. For example, a user may download a web page from a server, wait some time as they browse the page, then download another page from the same server. In a circuit switched network the user would either need to reserve a circuit for the entire duration of time it takes to download both pages, or the user could establish two circuits, one for each page. In the first case, network bandwidth is reserved for the connection during the idle time when the user is downloading a page resulting in low network utilization. The second case makes more efficient use of network resources, but each connection must incur the overhead and delay of establishing a circuit. If the amount of time required to set up the connection is comparable to the amount of time required to transfer the data (i.e. for connections which transfer small amounts of data), this approach is quite inefficient.

To better handle bursty data traffic, the ARPANET used a technique known as *packet switching*. In a packet switched network, data to be transmitted is broken down into small units called packets. A header containing the destination address, source address, and other control information is added to the packet, and the packet is transmitted into the network. The packet is forwarded from one link to another along the path from the source to destination.

Packet switching allows users transmitting bursty traffic to efficiently share network resources. The disadvantage of a packet switched network, however, is that packet switching only provides *best effort* service. In a circuit switched network, once a connection has been established, the user is guaranteed they will receive the amount of bandwidth reserved by the connection. No such guarantees are made in a packet switched network. A burst of packets from a large number of uses may arrive to a link at a rate faster than the link can support. As a result, the packets must be buffered at that link until they can be transmitted. This can result in delays as the packets are waiting to be transmitted, and if the buffer in the node completely fills up, packets may be lost. While this may seem undesirable, traditional data applications can tolerate such delay and loss. For example, if a packet of an email message experiences 400 ms of network delay, the recipient is unlikely to notice. Furthermore, even if a packet is lost in the network, protocols such as TCP provide a mechanism to detect the loss and retransmit the packet.

The ARPANET was quite successful and grew to nearly 100 nodes by 1975. As a result of the success of the ARPANET, DARPA began exploring the use of packet switching over satellite networks (SATNET) and wireless radio networks (PRNET and

Alohanet). Each of these networks, however, used a different set of protocols, packet formats, and address conventions to identify source and destination systems. To allow users to communicate across multiple networks, a set of protocols was developed to interconnect, or internet, these networks [17]. These protocols were later standardized as the Transmission Control Protocol (TCP) [96], and the Internet Protocol (IP) [95].

In addition to DARPA, many other agencies were impressed by the success of the ARPANET, and they decided to develop their own networks. In the mid 1970s, the Department of Energy created HEPNET to connect the high-energy physics community. In 1981, the National Science Foundation (NSF) funded CSNET to connect computer science departments. Corporate entities such as IBM, Xerox, and Digital Equipment Corporation also developed large data networks for their own use. To support users without sufficient resources to deploy their own dedicated networks, commercial network providers developed networks to which users could purchase connections. The first of these commercial networks was Telenet, developed by BBN [22] in 1974.

In 1985, NSF began developing a network known as the NSFNET to connect all academic users, regardless of discipline [66]. The architecture of this network was somewhat different from that of the ARPANET. The ARPANET was a monolithic network administered by a single entity under DARPA contract. To connect to ARPANET, a research group needed to be funded by DARPA and received a network connection as part of the contract. The NSF established a similar backbone network, the NSFNET backbone, to connect five supercomputer centers located at Cornell, the University of Illinois, the Pittsburgh Supercomputing Center, UC San Diego, and Princeton as well as the National Center for Atmospheric Research in Boulder, CO. However, the NSF also funded separate regional networks which connected universities and other research institutions within a common geographic area to the NSFNET backbone. This led to the creation of a number of regional Internet Service Providers (ISPs) such as NYSERNET, JVNCNET, and SDSCNET. Most users would connect to these regional networks rather than connecting directly to the NSFNET backbone itself.

To support growing traffic demand the NSFNET backbone was expanded to 13 nodes in 1988. The additional nodes included Merit Networks at the University of Michigan, which administered the backbone; BARRNET in San Francisco; MIDNET in Lincoln, Nebraska; WESTNET in Salt Lake City, Utah; NWNET in Seattle, Washington; SESQUINET at Rice University; and SURANET at Georgia Tech. The NSFNET backbone was also upgraded from 56 kb/s links which had been used in the ARPANET (and which are the same speed as dial-up modems today), to 1.5 Mb/s T-1 links and later 45 Mb/s T-3 links.

The aim of the network was to support academic research, and consequently the original NSFNET charter prohibited the use of the network for commercial purposes. As a result, many regional ISPs transitioned to private companies (rather than receive funding from NSF) and began offering commercial network service in addition to connection to the NSFNET. For example, NYSENET became a commercial network known as PSINET. This also led to the development of networks such as UUNET

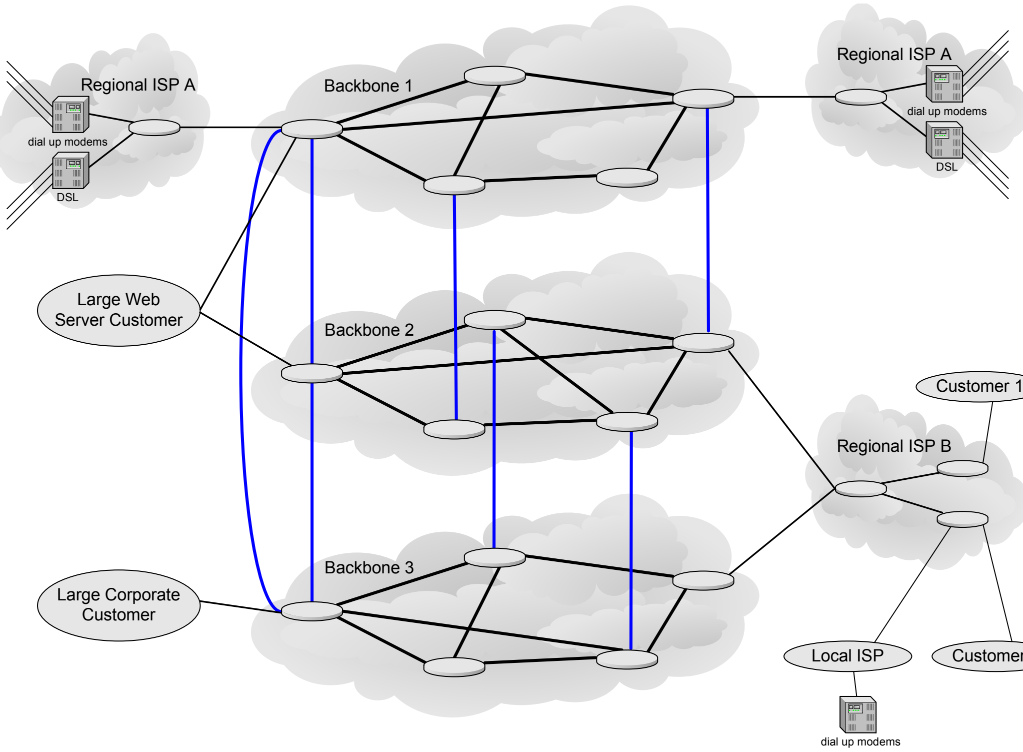


Figure 1.1: Internet architecture

which were commercial equivalents to the NSFNET backbone. As these commercial networks grew, NSF subsidy of the NSFNET backbone became unnecessary, and in April 1995 the NSFNET was retired. The remaining combination of commercial backbone and regional networks is the underlying infrastructure of the Internet today.

A conceptual diagram of the Internet architecture is shown in Figure 1.1. The Internet is composed of a small number of Tier-1 backbone networks (e.g. MCI/UUNET, Sprint, AT&T, and Qwest) that provide connectivity over a wide geographic area. Most of these networks also provide international connectivity.

Connected to the Tier-1 networks are smaller Tier-2 and Tier-3 networks which purchase transit service from one or more Tier-1 networks. Examples of such networks include large nationwide dial-up service providers such as AOL or Earthlink. These networks will establish a dial-up modem, DSL, or cable-modem pool for each city in which they provide service, and purchase a connection from the modem pool to one or more backbone networks. Similar ISPs, such as RCN, offer higher speed connections (e.g. 45 Mb/s DS-3) for business. Some very large corporations or web servers, which require very high bandwidth connections (e.g. 155 Mb/s OC-3), may purchase service directly from the backbone itself.

As part of the transit service agreement, the backbone network agrees to accept traffic from the customer and deliver it to any destination in the Internet. To reach all destinations, the Tier-1 networks must therefore be interconnected. These interconnection locations are known as *peering points*. For performance reasons multiple peering points may exist between two Tier-1 networks at different locations.

1.2 Application Requirements

As the network architecture has evolved, so have the applications which use the network. As mentioned above, the initial goal of the ARPANET was to provide remote access to large time-sharing computers. In fact, the first message transmitted between two nodes in the ARPANET was a command to login to the host at the Stanford Research Institute from a host at UCLA [66]. The remote login application, however, was not responsible for the rapid growth of the network. Instead, the growth was initially driven by the use of email, which was first developed in 1972 [114]. By the following year, 75% of the traffic on the ARPANET was email [22].

For the next 20 years, email, network news (a system of online discussion groups), and file transfers were the dominant use of the network. In the 1990s, the World Wide Web (WWW) replaced these applications as the dominant source of network traffic [44]. The Hypertext Transfer Protocol (HTTP), the protocol used to transfer web pages, was originally developed by Tim Berners-Lee for the European particle physics research center CERN in 1991 [6]. It was developed so that CERN researchers could easily access the work of other research groups. A text-mode WWW browser was released in 1991, and over the next two years several graphical browsers were developed. The most popular of these was Mosaic which was developed in 1993 by a group of programmers at the National Center for Supercomputing Applications at the University of Illinois (Mosaic later evolved into the Netscape browser). The graphical interfaces were popular, and by 1995, the WWW was the dominant network application [44]. Today, the WWW is still the dominant source of network traffic in most networks. However, as we will see in Chapter 2, peer-to-peer file sharing applications are beginning to dominate traffic volume on some links. Examples of such applications include Napster, Kazaa, and Morpheus.

The growth of the WWW and real-time applications has led to a fundamental change in the requirements for the network. For applications such as email, file transfer, and network news, the best effort service provided by a packet switched network is sufficient. These applications use a protocol known as TCP to detect and retransmit lost packets. Furthermore, when TCP detects a loss, it reduces the transmission rate of the application in order to alleviate the congestion in the network which presumably caused the loss. This procedure works well for non-interactive applications such as email. Applications such as the WWW, real-time video, voice, and online gaming, however, are interactive. Users expect that the transmission of data through the network occurs within some bounded period of time. For WWW applications, users expect a page download will take no more than 5-10 seconds [10]. It is possible to recover from a few lost packets and still meet this expectation, but during periods of heavy congestion, page downloads can take much longer than several seconds [84]. Voice communication has much more stringent delay requirements. Studies have been performed in which users were asked to rank the quality of voice telephone calls with different levels of delay [52]. For calls in which the delay between the time at which one user speaks a word and the other user hears it is only 50 ms, 90% of the calls received a "good" ranking. When the delay increased to 150 ms, 88% of the calls received a "good" ranking. With a delay of 250 ms, only 80% of the calls received a "good" ranking.

While real-time media applications such as voice and video have strict delay requirements, they can tolerate a small number of packets which exceed these delay requirements. Such packets will arrive too late to be played back, and will simply be considered to be lost by the receiver. The effects of these losses can be minimized using error concealment and correction techniques [99].

In addition to real-time applications, some data applications also require better than best effort service from the network. The WWW and other network applications have become mission critical for many businesses. This includes business which communicate directly with customers over the network (e.g. electronic commerce companies and online stock trading companies), as well as business which establish Virtual Private Networks (VPNs) between different sites in order to facilitate communication with the company.

1.3 Supporting Delay Requirements

To support business customers and delay sensitive applications, ISPs offer Service Level Agreements (SLAs) which specify bounds on the loss and delay that a customer's traffic will experience within the provider's network. Ideally, these SLAs would specify the loss and delay performance experienced between an ISP's customer and any destination in the Internet. However, this type of SLA is quite difficult to support since the traffic passes through many networks. Consider, for example, *Customer 2* who is connected to *ISP B* in Figure 1.1. This customer has an audio file which they would like to stream to other users. If the user is a dial-up modem customer connected to *ISP A*, the audio traffic will pass through four networks: *ISP B*, *Backbone 2, Backbone 1*, and *ISP A*. If the user is part of the *Large Corporate Customer*, then the traffic will pass through two networks: *ISP B* and *Backbone 3*. In order for *ISP B* to offer an SLA which guaranteed a certain delay between *Customer 2* and any destination in the Internet, all networks would need to support the required level of performance. This would require the cooperation of the more than 10,000 independent networks which comprise the Internet [106].

Since it is not possible for a network provider today to offer a delay guarantee to any destination in the Internet, the service provider only offers an SLA which guarantees the performance within their own network. Such SLAs offer either *deterministic* guarantees or *probabilistic* guarantees. Deterministic guarantees provide hard bounds on the performance of the network, e.g. the delay through the network will always be less than 30 ms. Probabilistic guarantees provide statistical performance bounds, e.g. 99% of the packets will experience less than 30 ms delay. The challenge facing network providers is how to design their networks to meet these SLAs.

There are two basic approaches which may be used to meet the delay-based SLAs. One approach, known as traffic differentiation, is to provide preferential treatment to latency sensitive traffic. This approach can result in efficient utilization of network resources, but it adds complexity and cost to the network. Below we describe two mechanisms, IntServ and DiffServ, which have been proposed to implement traffic differentiation in IP networks. A second approach, known as bandwidth provisioning, is to provide sufficient bandwidth in the network so that all traffic meets the most stringent delay requirements. This approach is simpler, but may be costly in terms of the bandwidth required.

1.3.1 Integrated Services (IntServ)

IntServ is a mechanism by which circuit functionality can be added to the packet-switched Internet [12]. When a user wishes to communicate with a host, the user transmits a reservation request message that specifies the delay performance the user desires and the characteristics of the traffic the user will transmit. Network elements (i.e. routers) determine if enough bandwidth and buffer space is available to meet the performance requested by the user. If so, the connection is accepted and the resources are reserved for that connection. If not, the network denies the connection request. The protocol which is used to make the connection requests is known as RSVP [13].

One of the key components of this approach is the manner in which the user's traffic is specified and the procedure which routers use to determine if sufficient resources are available to support the performance requirements of the traffic. Since data applications are inherently bursty, and this burstiness can result in queuing delay, the traffic specification must include more information than simply the average data rate. One option is to specify a long-term average rate and a peak data rate [41]. With such a specification, deterministic guarantees can be met by providing enough bandwidth so that the desired performance level is satisfied when all users are transmitting at the peak rates. A slightly more sophisticated traffic characterization is the token bucket. The token bucket specifies a long term average token rate and a bucket size. Tokens accumulate in the bucket at the average rate, and a maximum number of tokens can be held in the bucket. When a packet is to be transmitted, the router checks if a token is available. If so the packet is transmitted and a token is removed from the bucket. If not, the packet transmission is delayed until a token is available. Using the procedure described in [26],[27] , routers are able to determine the delay that will be experienced by traffic conforming to a particular token bucket specification and can therefore determine if a connection should be accepted or rejected. This type of analysis has been extended to handle more elaborate types of deterministic traffic specifications which specify the peak arrival rate over multiple time intervals [11].

Many applications, however, do not require deterministic guarantees. Voice, audio, and video, for example, can tolerate the loss of some packets. For such traffic statistical guarantees are sufficient. To model such traffic, a statistical estimator known as the *effective bandwidth* has been developed [55],[45],[60],[49],[30]. The effective bandwidth is a measure of the amount of capacity required in order to support a particular delay requirement for a flow.

One difficulty with all of these descriptors is determining the parameters for a given flow. For some flows, such as those streaming a stored video file or an audio music file, the traffic descriptors can be computed before transmission begins based on the file characteristics. However, for many types of network traffic it is difficult to determine, a priori, the characteristics of a particular flow. For example, one cannot predict the characteristics of the transmission of a web page since the transmission is controlled by TCP and will be affected by the amount of bandwidth and the amount of loss experienced in the network. To meet delay guarantees for such systems, measurement based algorithms have been developed [46],[58],[48]. In such systems, network entities measure the traffic to estimate the traffic characteristics. When these systems detect the network can no longer support any new traffic, new connection requests are denied.

The connection admission control mechanism is a critical aspect of the integrated services architecture. Equally important are the scheduling mechanisms which allocate network resources to the admitted flows. The scheduler is the part of the router which is responsible for selecting the order in which packets are transmitted on a link. The simplest scheduler is a first-in-first-out (FIFO) scheduler. Packets are inserted into the queue in the order in which they are received, and they are removed from the queue in the same order. However, this type of scheduler may not be able to meet delay requirements. For example, if one user transmits a large burst of data, all packets arriving after the burst will be delayed. To meet delay guarantees it may be beneficial to serve the packets in a different manner. One option is to perform strict priority scheduling. In the simplest case, a router maintains two queues, one for high priority data and one for low priority data. Packets in the low priority queue are only served if there are no packets waiting in the high priority queue. While this guarantees low delay performance for the high priority queue (assuming limited amounts of high priority traffic are admitted to the network), it is possible that packets in the low priority queue never get served. In many cases it is desirable to allocate a minimum amount of service time to each queue. With such a system, each user can be assigned their own queue and allocated sufficient bandwidth to meet their delay requirements. Algorithms such as Weighted Fair Queuing (WFQ) [31] and Packetized General Processor Sharing [87],[88] have been proposed to accomplish this goal. These algorithms, however, are quite difficult to implement, so many approximation algorithms have been developed. Readers interested in a discussion of these algorithms, as well as other fair queuing algorithms can find a summary in [115] and [50].

The primary difficulty with IntServ is that each flow must establish a reservation with and be treated individually by each router through which it passes. In Chapter

2 we will see that some high speed backbone links can carry up to 300,000 active flows in a one minute interval. While it is possible to implement schedulers which serve this large number of flows, processing the connection setup messages is quite difficult. To address this problem, several protocols to aggregate RSVP requests from multiple users into a single request have been proposed, but none were standardized. In addition, a unique distributed scheduling algorithm known as Dynamic Packet State (DPS) has been developed [105]. In DPS, edge routers (which process a relatively small number of flows) attach a header to each packet indicating how the packet should be scheduled within the backbone network. Using only the header information, core routers can make the appropriate scheduling decisions without knowing the details of each flow in the network.

1.3.2 Differentiated Services (DiffServ)

A second approach to implementing service differentiation is to offer a single service for all packets with similar performance requirements. For example, voice traffic from all users may be aggregated into a single class. This approach was developed to address the scalability problems of IntServ. In this approach, the network only maintains queues for each traffic type, rather than for each user, and network routers do not need to process connection requests for individual user flows.

This approach has been standardized as Differentiated Services (DiffServ) [9]. In DiffServ, the ISP and the customer establish a contract known as a Service Level Agreement (SLA). The SLA specifies a traffic profile which the customer's traffic must meet and the service that the traffic will receive. As long as the customer's traffic fits the profile, it will receive the promised service. If the traffic exceeds the profile, however, it can be dropped or classified into a lower quality service.

Networks which implement DiffServ operate as follows. When a packet enters the network, the ingress router classifies the packet according to the SLA established with the customer, and marks the packet according to the class of service it should receive. At each node in the network, the router treats packets according to the *per-hop behavior* (PHB) defined for the traffic class to which the packet belongs. Two basic PHBs have been defined. The Expedited Forwarding PHB states that a minimum amount of bandwidth is to be reserved at each router for the EF traffic class [29]. The amount of bandwidth reserved is a function of the traffic volume and the delay guarantee which is offered by the network. A second PHB is the Assured Forwarding (AF) PHB [51]. In this PHB, four AF classes are defined, each of which receives a configurable amount of network bandwidth. Within a single AF class, there exist three levels of drop priority. These drop priorities are intended to provide further service differentiation, and can be used for such applications as layered video [62].

1.3.3 Bandwidth Provisioning

An alternative to performing traffic differentiation is bandwidth provisioning. This approach is based on the observation that it is always possible to meet a certain delay requirement by simply providing enough bandwidth in the network. However, due to the bursty nature of data traffic, the bandwidth requirements may be much greater than the average traffic volume. In fact, studies of traffic in access networks, have found that the network must have twice as much bandwidth as the average traffic volume in order to achieve an average queuing delay of less than five milliseconds for each link [35].

Despite its potentially high cost, many network providers use the bandwidth provisioning approach because of its simplicity. Using forecasts of expected sales volume and traffic growth projections, network engineers estimate how much traffic demand will increase before the next provisioning cycle. The network is then designed with sufficient capacity so that link utilization will not exceed some threshold (e.g. 50%) at any time before the next network upgrade. After the additional capacity has been installed, the network is monitored to determine if the link utilization is remaining below the threshold. If utilization on one link is consistently above this threshold, then the network operators will modify the routing protocols in order to distribute the excess traffic on other, more lightly loaded links. As we will see in Chapter 2, this approach is successful at achieving a network with minimal queuing delays.

1.4 Internet Backbone Networks

In the context of IP backbone networks, two factors make the bandwidth provisioning approach attractive. First, there are costs associated with traffic differentiation. While some of this cost is related to the additional complexity required in network routers, much of the cost is associated with the management and operation of the network. Installers must be trained to configure the traffic differentiation mechanisms when routers are installed in the network, network operators must be trained to manage the different traffic classes, and the operators must be able to troubleshoot the various traffic differentiation mechanisms when problems occur.

A second factor which makes bandwidth provisioning attractive is that traffic differentiation may not provide much benefit in backbone networks. A goal of packet switched networks is to aggregate traffic from many users so as to reduce the amount of resources required in the network. Consider two users who transmit data at an average rate of 1 Mb/s, but over short periods of time can transmit at a peak rate of 2 Mb/s. If each of these users were to reserve a dedicated circuit for their communication, each would need to reserve 2 Mb/s of bandwidth. However, it is unlikely that both users will be simultaneously transmitting at their peak rates. Therefore, if the traffic from both users is combined, it may be sufficient to provide only 3.5 Mb/s, rather than 4 Mb/s for the aggregate traffic. There will be occasional periods when both users do simultaneously transmit at their peak rates and some queuing delay will be experienced.

Traffic in backbone networks is aggregated from between 10,000 and 300,000 users. As a result of the high degree of aggregation, as well as due to the low packet transmission times (a 1500 byte packet takes only 5 μs to transmit on a 2.5 Gb/s OC-48 link), it is expected that queuing delays in backbone networks will be low, and traffic differentiation may not provide much benefit.

1.5 Thesis Contributions

This thesis studies the feasibility of the bandwidth provisioning in the context of Internet backbone networks. In particular, we develop a procedure to determine the bandwidth required in an Internet backbone network to meet a constraint on the end-to-end delay which is experienced in the network.

The bandwidth requirements are dependent on the characteristics of backbone traffic. The characteristics of such traffic, however, are not currently well understood. Nearly all work on traffic analysis has used measurements of traffic whose average rate was between 200 kb/s and 10 Mb/s. At the time these measurements were collected, they represents very large traffic volumes. Links in today's backbones, however, carry traffic volumes which exceed 1 Gb/s. The first contribution of this thesis is the design and deployment of a traffic measurement system which collects detailed packet-level measurements from high-speed backbone links. This system is deployed in the Sprint IP backbone, a commercial Tier-1 network. Measurements collected with this system are used for this study as well as for a wide range of research projects undertaken by the Sprint Advanced Technology Labs. The architecture of the measurement system and observations on backbone traffic are presented in Chapter 2.

The measurements confirm that backbone traffic is aggregated from a large number of users, and that the average rate of these users is much smaller than the total link capacity. Only 1% of users ever exceed a rate of 500 kb/s, even on 2.5 Gb/s backbone links. As a result of this large degree of aggregation, we expect that the statistical properties of the traffic arrival process should have Gaussian distributions. From the measurements we find that this is the case for backbone traffic and that it can be fully described by Gaussian distributions. The second contribution of this thesis is a traffic model which we call *two-scale Fractional Brownian Motion* (FBM) which captures the observed Gaussian behavior of backbone traffic. This is an extension of traditional Fractional Brownian Motion which was originally proposed as a model for network traffic in [83],[112]. In Chapter 3 we present this model and derive expressions for the delay distribution of queue fed by two-scale FBM. We also develop a procedure to compute the end-to-end delay through a network where the aggregate network flows are modeled with two-scale FBM.

This finding seemingly contradicts earlier work on traffic modeling which found that network traffic is quite complex, especially when looked at over time intervals of less than 100 ms, and therefore second-order Gaussian models can be quite inaccurate [38]. However, as mentioned above, [38], as well as most other traffic measurement studies, consider traffic with an average rate of less than 10 Mb/s. In this thesis we find that for highly aggregated traffic (greater than 50 Mb/s), and for some moderately aggregated traffic (between 5Mb/s and 50Mb/s), the traffic arrival process at time scales between 1 ms and 100 ms is Gaussian. We also find, consistent with earlier measurements, that traffic with low aggregation (less than 5 Mb/s) cannot be described using second-order models.

Using the two-scale FBM model, we can develop several procedures to evaluate the bandwidth required to support a particular end-to-end delay constraint. One approach is to derive a constraint on the delay for a single link in the network based on the total end-to-end delay which is allowed (e.g. divide the total end-to-end delay evenly among all links on a path through the network). Using the two-scale FBM model, one can determine the amount of bandwidth required on each link to meet such delay constraint. Using a set of over 300 one-hour traffic measurements, we derive two-scale FBM model parameters for "average" backbone traffic as well as for the "most variable" backbone traffic. Using these models we determine the maximum utilization at which backbone links may be operated and still satisfy various delay constraints. However, determining how to divide the end-to-end delay among the different links in the network can be quite difficult. We therefore develop two algorithms which consider the entire network and determine the bandwidth required on each link in the network so that the end-to-end delay constraints are satisfied. Finally we consider the problem of rerouting traffic to take advantage of excess network capacity which is available in cases where it is not possible to add network bandwidth in a timely fashion. The algorithms and results are presented in Chapter 4.

Chapter 2

Internet Backbone Traffic Measurements

2.1 Introduction

Most work on network traffic modeling and analysis has been based on traffic measurements collected in the early 1990s from links between universities and the Internet or links between large corporate research labs and the Internet. These measurements were collected at Bellcore [67], the University of California Berkeley and the University of Southern California [28], the Lawrence Berkeley Laboratory [92], an academic network between the United States and the United Kingdom [111], at DEC Western Research Laboratory, and from the coNCert network in North Carolina. At the time these measurements were collected, they represented high volume WAN traffic and were the best data available. However, these measurements are not representative of traffic in today's commercial backbone networks. Commercial backbones carry much larger volumes of traffic (10s of Mb/s to 10s of Gb/s rather than the 200 kb/s to 10 Mb/s observed in the prior measurements), and the applications that generate the traffic in today's network are much different than the applications used in the early 1990s.

Unfortunately, traffic measurements from commercial backbone networks are not easy to acquire. Unlike the NSFNET backbone which had extensive monitoring facilities [20], commercial networks frequently only collect measurements of the average traffic volume. Furthermore, measurement data is often proprietary and not released for academic research.

Some measurement data from commercial backbone networks is available. Data from the MCI backbone has been published in [108] and [19]. These measurements consist of information about the individual user flows observed on several links in the MCI network. This information is sufficient to study the types of applications which generate network traffic and the characteristics of user flows. Flow measurements, however, do not provide information about the packet arrival process which is needed to study queuing delay and related network design problems. Some packet-level measurements are available from the NASA Ames Internet Exchange [74], and from the NLANR Passive Measurement and Analysis project [75], but the duration of these measurements is only 90 seconds to several minutes.

In order to obtain traffic measurements needed to support various research projects, the Sprint Advanced Technology Labs began developing the IPMON measurement facility in 1999. The goal of the IPMON facility is to collect traffic measurements from multiple locations in the Sprintlink IP backbone network in order to study:

- Characteristics of backbone traffic - traffic composition in terms of applications and protocols, characteristics of individual TCP flows, packet arrival patterns.
- Network performance - delay and loss experienced through the Sprint network, as well as end-to-end delay and loss performance that can be inferred from traffic observations.

Three measurement techniques were considered for the design of the IPMON system:

1. Active measurement systems
2. Flow-level measurement systems
3. Packet-level measurement systems

Active measurement systems transmit probe traffic into the network and measure the loss and delay performance of these probes. This provides a good mechanism to evaluate average network performance, but measuring rare events, such as the longest 1% of delay experienced by packets, requires a large amount of probe traffic.

Flow-level measurement systems collect information such as the start time, duration, number of packets, and number of bytes for each individual user flow in the network. However, as mentioned above, flow-level measurements do not provide information about the packet arrival process which is needed to study queuing delays.

Furthermore, flow-level measurements do not provide information about network performance.

Packet-level traffic measurements record the header and timestamp of every packet transmitted on a link in the network. This type of measurement provides the most detailed view of network traffic. The drawback to such systems is that they collect a much larger volume of data than the other two approaches. For a given amount of storage, flow-level systems or active measurement systems can collect measurements for a much longer period of time than packet-level measurements. However, it is possible to provide enough storage in a PC-based measurement system to collect several hours of data which is sufficient to study the problems in which we are interested. We therefore decided to use the packet-level measurement approach.

The remainder of this chapter describes the architecture of the IPMON facility and uses measurements collected in August 2000, September 2001, and April 2002 to characterize traffic in the Sprint IP backbone. Section 2.2 reviews other network traffic measurement systems and motivates the need to design new measurement facility. Section 2.3 describes the details of the measurement system. Section 2.4 performs a high level analysis of the traffic. A detailed analysis of the packet arrival characteristics is postponed to Chapter 3.

2.2 Existing Measurement Systems

Many systems have been developed to collect network traffic measurements. These systems can be classified into three general categories: packet level measurement systems, flow level measurement systems, and network performance measurement systems.

Packet level measurement systems are designed to collect detailed information about each packet transmitted on a single link in the network. These systems can be as simple as a computer running *tcpdump* [57] connected to a shared Ethernet segment. *tcpdump* records the header information of packets received on the computer's network interface and the time at which each packet was received. It can be configured to record the header information for all packets observed on the link or only for packets whose header matches a certain pattern (e.g. all packets with a particular destination IP address). Measurements collected by such systems have been used in many prior traffic analysis studies such as [90]. AT&T has also developed a *tcpdump* based measurement systems known as the PacketScope for FDDI and T3 links [2].

A shortcoming of the *tcpdump* based systems is that the packet timestamps are only as accurate as the system clock of the computer which is used to collect the measurements. To improve the accuracy of the system clock, the Network Time Protocol (NTP) is used to synchronize the computer system to UTC (Coordinated Universal Time) [78],[80]. The details of the operation of NTP are presented later in this chapter, but in general, NTP can synchronize the system clock to within tens of milliseconds of UTC. Adding dedicated clock synchronization hardware and modifications to the operating system can provide synchronization to within tens of microseconds of UTC [79]. However, there is still an undetermined amount of error in the timestamp, since the timestamp is recorded when the operating system receives the packet from the network interface, not the time at which the packet was actually observed on the link. The packet spends some period of time in the receive buffer on the network interface card.

To improve the timestamp resolution, several custom packet measurement systems have been developed. These systems require dedicated hardware which timestamps the packets immediately when they are received at the network interface. These systems include an Ethernet packet level measurement system [68] whose data has been used in [67],[112], an FDDI packet level measurement system whose data has been used to study user billing policies [33], and a measurement system for OC-3 and OC-12 ATM links known as OCxMON [3],[108],[19]. The OCxMON systems are very similar to the IPMON system, but they have two primary differences. At the time when the IPMON systems were deployed, the OCxMONs only supported ATM networks. The Sprint network uses the Packet over Sonet (PoS) protocol rather than ATM, so the OCxMON could not be used in our network[1] . Furthermore, the OCxMON are only capable of recording the packet data in main memory of the measurement system, and they are unable to store the data to disk. As a result, the systems can only collect several tens of minutes of data on a moderately loaded OC-3 or OC-12 link. The IPMON systems have enough processing power to record the packet data directly to disk and can record data for several hours.

Flow-level measurement systems record information about each user flow observed

[1]After deployment of the IPMON system, the OCxMONs (now known as CoralReef) were modified to support POS.

on a link in the network. Typically a flow is defined as a sequence of packets transmitted between the same source and destination IP address which have the same IP protocol number and source and destination TCP/UDP port numbers. The information collected for each flow typically includes the flow start time, the flow duration, and the number of bytes and packets transmitted in the flow. The systems which collect flow level statistics are the OCxMON systems [108],[19],[74] (these systems can be configured to collect either packet or flow level measurements); NeTraMet [14], an implementation of the Internet Engineering Task Force's (IETF) Realtime Traffic Flow Measurement system [15]; and Netflow [81], a capability built in to Cisco routers which allows the routers to collect and export flow information to an external data collection system. Netflow data can be analyzed using an application known as *cflowd* [76]. AT&T has developed an extensive measurement system to collect and analyze Netflow data from their network [16],[39],[40].

Packet-level and flow-level measurement systems provide information on the traffic carried by the network, but they do not provide information about network performance. To measure the loss and delay performance of a network, systems are designed to inject probe traffic into the network and measure the performance seen by the probe traffic. These systems are frequently referred to as *active* measurement systems since they actively transmit data into the network. Flow and packet level measurement systems are referred to as *passive* measurement systems since they just observe network traffic.

Network performance measurement systems include the Network Probe Daemon

(NPD) [93],[94] which has evolved into the National Internet Measurement Infrastructure (NIMI) [91],[89], the PingER system which uses *ping* messages to monitor network performance between high energy particle physics research labs [73], and the RIPE test traffic measurement project, which measures bandwidth and delay performance between 24 sites in Europe, North America, and Israel [109].

These systems, however, have the same timestamp accuracy problems as described above for the *tcpdump* measurement systems. The timestamps are accurate enough to measure the round-trip time of network paths and they are capable of measuring losses, but they cannot measure one-way delays. The Surveyor project [59], on the other hand, has deployed a set measurement systems which are synchronized using the Global Positioning System (GPS). These systems are designed to measure one way delays through the network.

2.3 IPMON Architecture

The IPMON measurement facility is composed of three components. A set of monitoring entities which collect the packet trace measurements, a data repository used to archive the packet traces, and a data analysis system used to process the traces offline. This section describes the packet traces and each of these components in detail.

bytes	field description
0	64 bit
4	timestamp
8	record size (fixed to 64 bytes)
12	POS frame length
16	POS header
20	First 44 bytes of IP packet
⋮	
64	

Table 2.1: Packet record format

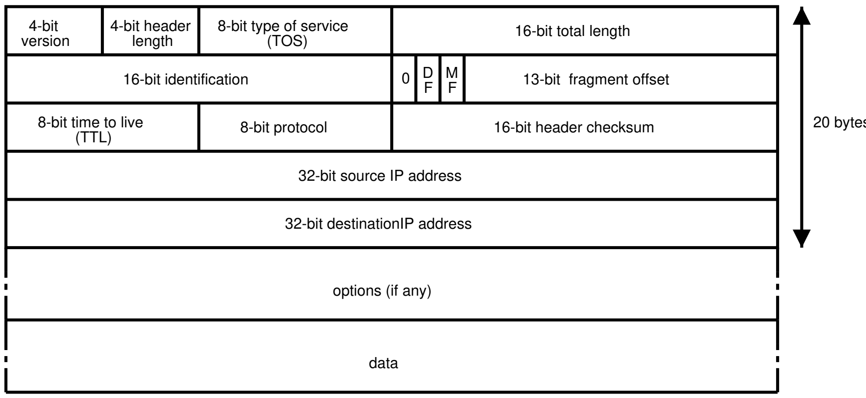
2.3.1 Packet Trace Format

A packet trace is a sequence of records which contain information about each packet observed on a link. The format of a packet record is shown in Table 2.1. Each packet record contains a 64-bit timestamp corresponding to the time at which the packet was observed on the link, a record size field, the length of the Packet-over-SONET (POS) frame which contained the IP packet, the POS header, and the first 44 bytes of the IP packet. The record size field is currently fixed to 64 bytes, and is included to support future systems with variable record sizes. The POS frame length is used to determine the actual size of the IP packet. The IP packet itself contains size information, but this information can be incorrect so the POS frame length is used to determine the actual packet size.

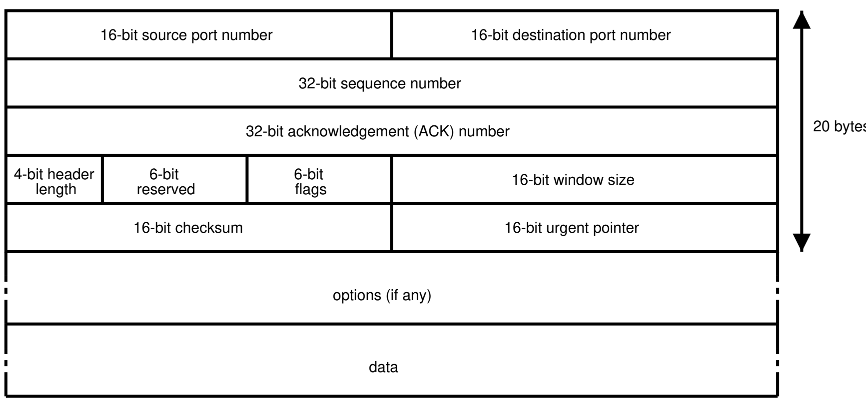
Packet-over SONET (POS) is the protocol used by Sprint and several other commercial IP backbones to carry IP network traffic directly over SONET optical networks. The POS protocol is specified in [56], and is similar to the PPP-over-SONET protocol described in [102],[103]. The protocol encapsulates a single IP packet into a POS frame. The framing consists of 4 bytes of header information and 2 bytes of trailer. The POS header contains a protocol field which is used to identify records which do not contain IP packets. These records correspond to "hello" messages and other data transmitted directly between the two routers.

The first 44 bytes of the IP packet contain the 20 byte IP header, as well as the 20 byte TCP header or 8 byte UDP header that is commonly used for most packets. The format of a standard TCP/IP and UDP/IP packet is shown in Figure 2.1. There are some packets which have extra header options in addition to those shown in Figure 2.1. For these packets, the headers may extend past the first 44 bytes which are recorded, and we may lose information. Fewer than .0003% of the packets we observe contain IP options, but 10% - 15% of the packets contain TCP options [42]. For the packets which contain TCP options, the first 44 bytes of the packet contain the standard TCP header fields, but not the optional fields.

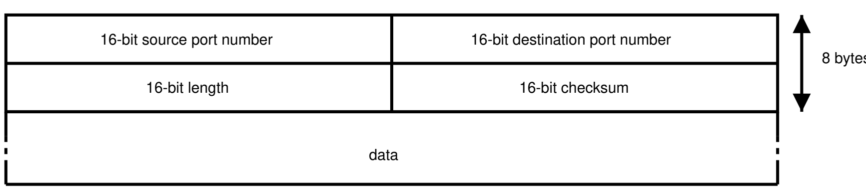
The packet traces can be used for many purposes. The IP headers contain information about the source and destination of the packet, and such information can be used to study how much traffic is flowing between different source/destination combinations. The TCP and UDP headers identify the application which generated the traffic, and can be used to study application characteristics. Information about individual TCP and UDP flows can be reconstructed by classifying to which flow each packet belongs. By analyzing the sequence numbers of the TCP flows, we can infer the round-trip-time of the TCP connections as well as the number of losses and retransmissions. To study network performance, the packet traces can be input into



(a) IP Header



(b) TCP Header



(c) UDP header

Figure 2.1: Packet header formats

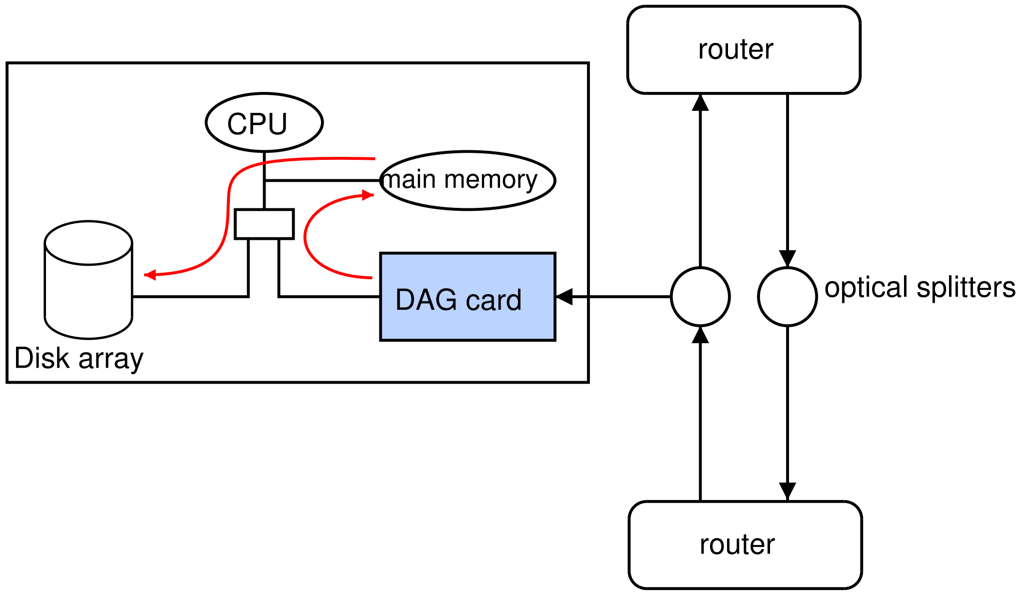


Figure 2.2: IPMON monitoring entity architecture

a network simulator. The results of the simulation can be compared to various traffic models in order to evaluate the accuracy of the models. Finally, the packet traces can be used to evaluate the delay performance of the existing network. If measurements are collected at multiple points, measuring the delay is simply a matter of comparing the timestamps of a packet as it is observed at the multiple locations . Using packet traces to measure network performance has two advantages over active measurement systems. First, the performance of actual user traffic is measured. We therefore do not need to determine the appropriate method in which to transmit the probe packets. Second, we do not need to worry about affecting network performance with probe traffic.

2.3.2 IPMON Monitoring Entities

The IPMON monitoring entities collect the packet traces. To collect packet traces, an optical splitter is installed on a single optical fiber which is to be monitored. It is important to note that a connection between two routers consists of two individual fibers, one for each direction. A separate splitter and measurement entity is needed to monitor each direction. The splitter creates a duplicate of the optical signal on the fiber, and this duplicate is input into an IPMON monitoring entity. The monitoring entities are a high-end Linux PC with a large disk array and a SONET network interface, known as the DAG card [21]. Two versions of the IPMON systems have been developed. One version is used to monitor OC-3 and OC-12 links, while a second version with a larger disk array and higher-speed internal system bus is used to monitor OC-48 links.

The IPMON monitoring entities operate as shown in Figure 2.2. The output of the optical splitter is connected to the DAG card which decodes the SONET payload and extracts the Packet-over-SONET (POS) frames. The DAG card extracts the first 48 bytes of each POS frame and adds a timestamp indicating the time at which the frame was received. The first 48 bytes contain 4 bytes of POS header and the first 44 bytes of the IP packet[2]. The timestamp and data are then transferred to main memory in the PC using DMA. The format of the packet record is shown in Table 2.1. If the IP packet contains fewer than 44 bytes (as is the case for small UDP and other non-TCP packets), the data is padded with all 0's.

[2]The DAG card processes data in 48 byte units since it was originally designed to operate in ATM networks which carry 48 bytes of user data in each ATM cell.

Once 1 MB of data has been copied to main memory, the DAG card generates an interrupt which triggers an application to copy the data from main memory to the hard disk. This may seem to be an inefficient process, the data is copied first to memory and then to disk. A more logical solution would be to copy the data directly to disk and bypass main memory. However, as described later in this section, the disks do not have enough bandwidth to be able to handle bursts of very small packets, and the DAG cards do not have enough on-board memory to buffer these bursts. It is therefore necessary to use main memory as a large buffer so as to not overload the hard disks. The DAG card collects data for a predetermined period of time (e.g. 24 hours), or until the disk space is exhausted. After the data collection is complete, the data is transferred to the lab for offline analysis.

There were five design issues addressed in the development of the IPMON monitoring entities:

- How to support monitoring 155 Mb/sec OC-3 links, 622 Mb/s OC-12 links and 2.5 Gb/sec OC-48 links.
- How to synchronize the timestamps generated by separate IPMON entities.
- How to minimize the physical space consumed by the monitoring entities.
- How to prevent unauthorized access to trace data.
- How to support complete remote administration.

Next we describe how each of these requirements are met in the system design.

	OC-3	OC-12	OC-48
link rate (Mb/sec)	155	622	2480
peak arrival rate (Mpackets/sec)	0.48	1.9	7.6
peak capture rate (Mb/sec)	248	995	3968
average capture rate (Mb/s)	49	199	793
1 hour trace size (GB)	11	42	176

Table 2.2: Data rate requirements

Monitoring High-speed Links

The primary difficulty with monitoring backbone links is the high rate at which the packet records must be stored to disk. The highest rate of packet arrivals occurs when a burst of very small packets arrives at the DAG card. Considering 40 byte packets (the minimum size TCP packet), the peak arrival rate is 0.48 Mpackets/s on an OC-3 link, 1.9 Mpackets/s on an OC-12 link, and 7.6 Mpackets/s on an OC-48 link. Since the DAG card creates a 64 byte record for each of these packets, the rate at which measurement data is generated ranges from 248 Mb/s for an OC-3 link to 3.97 Gb/s for an OC-48 link. Unfortunately, the RAID disk arrays used in the IPMON system do not have enough capacity to support this peak rate. The RAID array on the OC-3 and OC-12 systems can record data at a rate of 240 Mb/s (30 MB/s), while the RAID array on the OC-48 systems can record data at 720 Mb/s (90 MB/s). If the traffic on a network link was a constant stream of 40 byte packets, we would be unable to record packet traces. However, network traffic is a mix of packet sizes. In our measurements we find that the average packet size on a link ranges from 200 bytes to 500 bytes depending on the link (some links carry a large number of 40 byte TCP ACK packets while others carry a large number of 1500 byte TCP data packets).

Therefore, the average rate at which packet records are created is much less than the peak arrival rate. Considering an average packet size of 200 bytes, the RAID disk array only needs to have a bandwidth of 49 Mb/s for the OC-3 systems, 199 Mb/s for the OC-12 systems and 793 Mb/s for the OC-48 systems. The OC-3 and OC-12 systems are capable of capturing data at full line rate, but the OC-48 systems can only collect data up to 91% link utilization. Fortunately, the utilization of the OC-48 links in the network never reaches this value. These results are summarized in Table 2.2.

While the disks have sufficient bandwidth to record measurements at the average rate of the traffic, over short intervals of time, traffic can arrive at the peak rate. When this occurs, the data must be buffered until it can be recorded on disk. There is very little memory onboard the DAG card, so the data is buffered in main memory. This introduces another potential bottleneck in the system. The PCI bus which connects the DAG card to main memory as well as the main memory to the hard disk only has a bandwidth of 1056 Mb/s (132 MB/s) for the 33 MHz, 32 bit PCI bus in the OC-3 and OC-12 systems, and 4224 Mb/sec (528 MB/sec) for the 66 MHz, 64 bit PCI bus in the OC-48 systems. A single bus does not have sufficient bandwidth transfer data at the peak rate from the DAG card to main memory and at the average rate from main memory to disk using a single bus. We therefore use systems which have separate PCI busses for the DAG card and for the hard disk array. This architecture allows the IPMON monitoring entities to sustain data capture, even at OC-48 link speeds.

Timestamp Requirements

The second requirement for the IPMON monitoring entities is that the clocks which generate the packet timestamps must be synchronized between systems. With synchronized timestamps, we can measure network delay between two links by identifying a packet in a trace collected on the first link, identifying the same packet in trace collected on the second link, and compute the difference in timestamps.

This requirement is accomplished by synchronizing a 16 MHz clock on board the DAG card with a Global Positioning System (GPS) reference clock. The clock synchronization operates in the following manner as described in the DAG documentation [77]. The DAG clocks are initially loaded with the absolute time from the PC's system clock (e.g. 7:00 am Aug 9, 2000 PST). The PC system clocks are all synchronized to within 100 ms using NTP [80]. This guarantees that the initial time loaded into the DAG clocks is accurate to within 200 ms. However, 200ms is insufficient accuracy to measure network delays which can be less than 1 ms. Furthermore, the clock synchronization degrades over time. The crystal oscillators which generate the clock signal do not run exactly at 16 MHz. They will be slightly faster or slower depending on the temperature of the system and the quality of the oscillator. This clock drift causes the synchronization error to increase over time. To correct for this, a GPS receiver is installed at each monitoring site. The GPS receiver uses satellite signals to establish a reference clock which is synchronized to within 500 ns of UTC. The GPS receiver outputs a single clock pulse at the beginning of each second. This pulse-per-second (PPS) signal is distributed to the DAG cards in the IPMON measurement entities.

When the DAG card receives the first PPS signal after initialization, it resets the lower 24 bits of the clock counter (Note: 24 bits will count from 0 to 16 million - 16 MHz. If the lower 24 bits of the clock are all 0 it represents the beginning of a second). Thereafter, each time the DAG card receives the 1 PPS signal, it compares the lower 24 bits of the clock to 0. If the value is greater than 0, the oscillator is running a little bit fast and the DAG card decreases the frequency slightly. If the value is less than 0, the oscillator is running a little bit slow and the DAG card increases the frequency slightly. There is an initial period when the PPS is attempting to correct the initial clock skew, so we ignore the first 30 seconds of each trace to account for this.

There are several sources of error that may occur in the synchronization system. First, the GPS receivers at different locations may have up to a 500 ns difference in the PPS signals which they generate. Furthermore, it takes some amount of time for the PPS signal to travel along the cable between the GPS receiver and the DAG cards. The difference in propagation delay between the shortest and longest cable is approximately 30 ns. The final source of error is in the fact that synchronization happens only at 1 second boundaries. Between two PPS pulses, the DAG clock can accumulate some error. To test this aspect of performance, we measure the maximum clock error that is observed when the DAG card receives a 1 PPS interrupt. The maximum value we have seen in lab tests is 30 clock ticks which represents an error of 1.79 μs. The median error observed during these tests was 1 clock tick, or 59.6 ns. Combining these three factors, the worst case difference between any two DAG clocks is less than 2 μs.

While the DAG clocks are accurate to within 2 μs, there is another source of error in the timestamp generated by the OC-3 and OC-12 DAG cards. The packet timestamps are generated after the SONET data has been decoded. As the OC-3 and OC-12 cards were initially designed to operate on 53 byte ATM cells, data is transferred between the SONET framing chip and the chip which generates the timestamps in 53 byte units. In a Packet-over-Sonet network, this 53 byte unit may contain multiple packets. For example, it could contain one 40 byte packet as well as the first few bytes of a second packet. The DAG card considers both of these packets to have arrived simultaneously, when in fact their inter-arrival time was 2 μs for OC-3 links or 0.5 μs for OC-12 links. This results in an additional 2 μs of timestamp error. The OC-48 systems use a newer SONET framing chip which was designed to support POS directly and does not use 53 byte buffers. The cumulative effect of these errors is a maximum 4 μs of clock skew between DAG cards.

Physical Requirements

The IPMON monitoring entities are installed at operational network sites where physical space is limited. It is therefore important to minimize the amount of physical space which is needed by the IPMON measurement entities. For a network switch site, physical space is measured in terms of how much vertical rack space is used by the system. One unit (1U) of rack space corresponds to 1.75 inches.

The component in the IPMON measurement entities which takes up the largest amount of physical space is the hard disks. The challenge, therefore, is to achieve the optimal balance between physical size and hard disk capacity. The amount of data generated for one hour of trace measurements is shown in Table 2.2. Using a rack-optimized system, the OC-3 and OC-12 IPMONs are able to accommodate 108 GB of storage in only 4U of rack space. This allows the system to record data for 9.8 hours on a fully utilized OC-3 link or 2.6 hours on a fully utilized OC-12 link. The OC-48 systems have a storage capacity of 360 GB, but in a slightly larger 7U form factor. This is sufficient to collect a 2 hour trace on a fully utilized link. However, as we will see from the trace measurements, link utilization rarely exceeds 50% during the busiest hours of the day, and is frequently much lower during the nighttime hours. As a result, we are typically able to collect traces which last from 10 - 48 hours.

The amount of data collected in the packet traces is both the major advantage and the major disadvantage of the IPMON facility. Packet level measurements provide very detailed information about network traffic. However, it is not possible to collect such measurements from every link in the network, nor is it possible to collect measurements over days or weeks at a time. This is therefore not a practical solution for operational monitoring of an entire network. Instead, the goal of these traces is to provide researchers and network operators with a better understanding of network traffic.

Security Requirements

The IPMON measurement entities collect proprietary data about the traffic on the Sprint IP backbone. Preventing unauthorized access to this trace data is an important design requirement. To accomplish this, the systems are only accessible using two applications: *ssh* and NTP. *ssh* is an authenticated and encrypted communication program similar to *telnet* that provides to access a command line interface to the system. This command line interface is the only way to access trace data that has been collected by the system and to schedule new trace collections. *ssh* only accepts connections from a server in our lab and it uses an RSA key based system to authenticate users. All data that is transmitted over the *ssh* connection is encrypted.

The second type of connection accepted by the IPMON measurement entities is for NTP traffic. NTP synchronization is performed by establishing a broadcast NTP server on the LAN to which the IPMON measurement entities are connected. The entities only accept NTP messages which are transmitted as broadcast messages on a local network used exclusively by the IPMON systems.

Remote Administration Requirements

In addition to being secure, the IPMON systems must also be robust against failures since they are installed in remote sites which are not always accessible. To detect failures, a server in the lab periodically sends query messages to the IPMON systems. The response indicates the status of the DAG cards and of the NTP synchronization. If the response indicates either of these components has failed, the server attempts to restart the components. If the server is not able to correct the problem it notifies the system administrator that manual intervention is required. In some cases, even the *ssh* connection will fail, and the systems cannot be accessed over the network. To handle this type of failure, the systems are configured with a remote administration card that provides the capability to reboot the machine remotely. The remote administration card also provides remote access to the system console during boot time. In cases of extreme failure, the system administrator can boot from a write-protected floppy installed in the systems and completely reinstall the operating system remotely.

The one event that cannot be handled remotely is hardware failure. While it is possible to design a system with redundant hardware, such systems are quite costly. Since the monitoring systems play no role in the direct operation of the network, we decided hardware redundancy was unnecessary.

2.3.3 Data Repository

The data repository is a large tape library responsible for archiving the trace data. Once a set of traces has been collected on the IPMON systems, the trace data is transferred over a dedicated OC-3 connection from the IPMONs to the data repository.

A single 24-hour-long trace from all of the monitoring systems currently installed consumes approximately 1.2 TB of disk space (this will increase to 3.3 TB when the additional 20 systems are installed). The tape library has 10 individual tape drives which are able to write data at an aggregate rate of over 100 MB/sec. The rate at which the data can be transferred from the remote systems, however, is limited to 100 Mb/sec which is the capacity of the network interface cards on the IPMON systems. At this rate the raw data would take 26.4 hours to transfer from the IPMON systems to the tape library. To improve transfer time and decrease the storage capacity requirements, the trace data is compressed before being transferred back to the lab.

Using standard compression tools such as *gzip*, we are able to achieve compression ratios ranging from 2:1 to 3:1 depending on the particular trace characteristics. This reduces the transfer time to about 12 hours.

This transfer time presents another difficulty when exhaustively monitoring a network for operational purposes. An alternative solution would be to avoid the data repository and perform the analysis on the monitoring systems themselves. This is a good solution if there is a single type of analysis that is being performed on the traces. However, the data is used for many research projects, and some of the analysis performed in several projects requires multiple iterations through the trace. In addition, we would like to keep an archive of the collected data so that it may be used for future projects.

2.3.4 Analysis Platform

All data analysis is performed off-line by a cluster of 16 Linux PCs. The software tools used to perform the analysis are described below.

Single Trace Analysis Tools

This is the most general category of analysis in which a single trace is processed in order to analyze a particular characteristic of the network. This includes computing the packet size distribution or detecting denial of service attacks. Such types of analysis are performed using custom analysis programs developed to study the particular characteristic of interest.

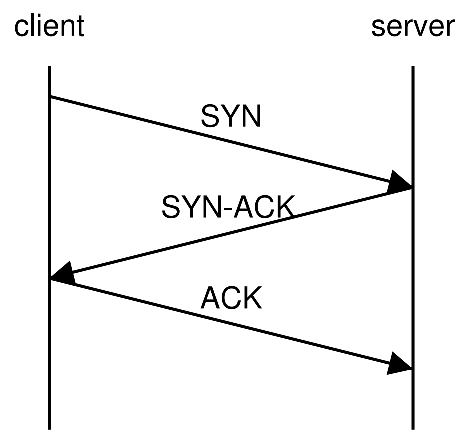


Figure 2.3: TCP 3-way handshake

Flow Analysis Tools

Flow analysis is performed in order to investigate the behavior of individual user connections. A user flow is defined as a sequence of packets with the same source and destination IP address, the same IP protocol (e.g. TCP or UDP), and the same source and destination TCP or UDP port number. This type of flow would represent, for example, the download of a web page or the transmission of an email message. The IP group at Sprint has written a utility which generates a set of records for every flow observed in a single trace. These records contain the source and destination IP address, the source and destination port, the IP protocol, the duration of the flow, the number of packets transmitted in the flow, and the total number of bytes transmitted in the flow. For TCP flows, it also records the number of TCP retransmissions, the number of out-of-sequence packets, the round-trip-time estimated for the flow, and the total amount of user data transmitted in the flow.

The round-trip time (rtt) of a TCP flow is estimated using a procedure similar to the one described in [54] and [71]. A TCP connection is initiated using the three-way handshake shown in Figure 2.3. The client transmits a SYN message to the server, the server responds with a SYN-ACK message immediately after receiving the SYN, and the client responds with an ACK message immediately after receiving the SYN-ACK. Since the responses are immediate (unlike ACK responses for data packets which are typically delayed up to 200 ms in order to "piggyback" multiple ACKs), the time between the initial SYN message and the ACK message which finishes the handshake provides a reliable estimate of the actual rtt of the connection. However, a single trace contains traffic only in one direction. For some flows we observe the initial SYN and the final ACK message. For other flows we only observe the SYN-ACK. We are only able to estimate the rtt for the flows in which we observe the initial SYN and final ACK message.

The number of retransmissions and out-of-sequence packets are determined by inspecting the TCP sequence number of each packet in the flow. If we observe a packet with the same sequence number as an earlier packet in the flow, it is considered a retransmission. If the sequence number of a packet is not the same as the expected TCP sequence number, we consider it to be out-of-order. It is important to note that these statistics only represent the number of retransmissions and out-of-order packets that we can observe in our measurements. There are retransmissions and out-of-order packets which we cannot detect. For example, if the source transmits a packet and it is lost before reaching the Sprint network, the source will retransmit the packet. However, we will not be able to detect this retransmission. Our measurements are therefore a lower bound on the actual number of retransmissions and out-of-sequence packets.

The total amount of user data transmitted is also determined using the TCP sequence numbers. The sequence numbers count the number of bytes transferred in the connection. Even though we only monitor traffic in one direction, we can use the sequence number of a packet and the ACK sequence number to determine the amount of user data transferred in both directions of the connection.

The flow analysis tool is very similar to the CoralReef tools developed by CAIDA [61]. We used CoralReef to validate the accuracy of our own flow analysis tool.

Timescale Analysis Tools

We have developed a tool to compute various statistics (e.g. number of packets, number of bytes) about the trace over small intervals of time. For example, this tool computes the number of bits that are transmitted every second or every minute for the duration of the trace. This type of analysis forms the basis for the traffic model discussed in Chapter 3, and will be discussed in detail in that chapter.

Delay Analysis Tools

If a packet is observed in a trace at time t and the same packet is observed in a second trace at time $t + \Delta$, then the delay incurred by the packet between the two monitored links is simply Δ. The key to measuring delay is to be able to identify an individual packet as it travels across multiple links in the network. The only three pieces of information that should change as a packet travels through the network are the TTL, TOS and checksum fields in the IP header. Using the remaining 40 bytes of data we collect for each packet, we can, in general, uniquely identify a packet.

To measure network delay, we developed a tool which identifies a unique packet in one trace and searches for the same packet in one or more additional traces. If the packet is found, the delay between the two links is computed by taking the difference between the timestamps. Due to the extremely large size of the trace, it was necessary to develop an efficient algorithm to perform this search process. Details of the search algorithm can be found in [86].

In some cases it is possible for two separate packets to have the same 40 bytes. In theory this should happen infrequently since the ID field for each packet generated by a particular source should be unique. In the traces we collect we do observe duplicate packets due to sources generating incorrect IP id fields or due to link layer retransmissions, but these packets only represent .001% to 0.1% of the total traffic volume [42]. In these cases, we typically ignore all packets which have duplicate values.

Tools for Identifying Traffic Subsets

A very useful property of packet traces is that certain packets can be extracted from the trace, and the characteristics of these subsets can be studied. For example, one could extract all packets which have a certain range of destination IP address, or one could extract all packets which correspond to web traffic. To facilitate such studies, we developed a program to identify all packets which match a certain criteria. These routines can be embedded in other analysis programs, or they can be used to create a separate packet trace containing only the selected packets.

Simulation Tools

Packet traces are also useful in that they can be input into a network simulator. We have written a simulator which reads multiple traces and simulates injecting the packets from the trace into a single FIFO queue. By combining many queuing simulations, one can simulate complex network topologies. Such simulations will be used later in this chapter and in Chapter 3 to evaluate how reducing the capacity of the network would affect queuing delay.

2.4 Measurement Results

In this section we characterize backbone network traffic in terms of the workload characteristics (traffic volume, application mix, etc.) and flow characteristics. We also report on the delay measured between multiple locations in the network. We begin by describing the architecture of the Sprint network and the set of packet traces which are used in the analysis.

2.4.1 Sprint Network Architecture

The Sprint IP network consists of a set of sixteen nodes known as Points-of-Presence (POPs) connected by high bandwidth OC-48 WAN (wide area network) links. Each POP has a two layer architecture shown in Figure 2.4[3]. At the highest layer, the

[3]Figure 2.4 only shows 15 of the 16 POPs in the Sprint network. It is intended to be a conceptual diagram of the Sprint network and does not necessarily reflect the exact network topology or

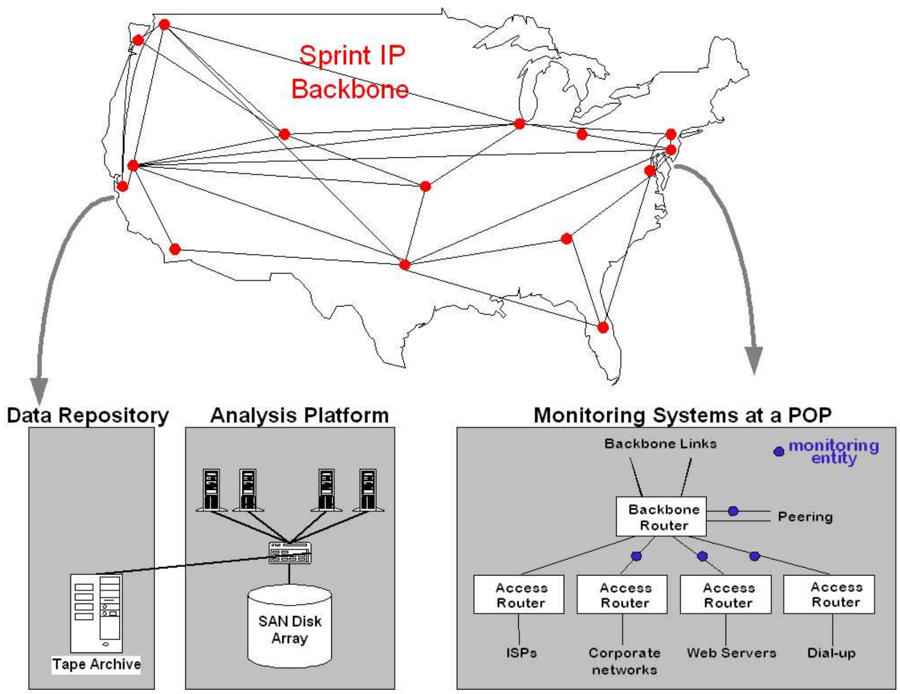


Figure 2.4: IPMON architecture

WAN links are connected to a set of backbone routers. At the lower layer, links to individual customers, ranging in speed from 1.5 Mb/s T1 links to 2.5 Gb/s OC-48 links, are connected to a set of access routers. Within the POP, the access and backbone routers are interconnected in a tree mesh topology. The IPMON monitoring entities monitor the links between the access routers and the backbone routers as well as peering links which connect the Sprint network with other Tier-1 networks.

2.4.2 Traces

The traces used in this study were collected on three different days, Aug. 9, 2000, Sept. 5, 2001, and Apr. 19, 2002. The Aug. 2000 measurements were all collected at

configuration.

a single location on the west coast of the U.S. All of the monitored links from that day are 155 Mb/s OC-3 links. The Sept. 2001 traces were collected from the same west coast POP, as well as from two POPs on the east coast. All of the monitored links in this day are 622 Mb/s OC-12 links. The Apr. 2002 traces were collected from 2.5 Gb/s OC-48 links in one east coast and one west coast POP. A summary of the characteristics of each trace is given in Table 2.3.

The names of the trace measurements use the following convention: *<POP_NAME>-<LINK#><LINK_DIRECTION>*. The first part of the name identifies the POP at which the measurements were collected. The second part of the name is a number which identifies a link. Note that the monitored links change from one measurement date to another. The last part of the name identifies the link direction. A single link between two routers consists of two physical fibers. The trace collected by a single measurement system corresponds to a single direction of a link. The traffic in the opposite direction is recorded in a separate trace. In general, we monitor both directions of each link. However, there are a few links for which we are able to only monitor a single direction due to optical power constraints. Occasionally, the optical power on one fiber will be much lower than the power on the opposite fiber. As a result, we are able to monitor only the high power link.

2.4.3 Workload Characteristics

In this section we present general characteristics about the aggregate traffic observed in each of the traces. Figure 2.5 shows the average traffic volume and the peak traffic

name	link type	start time	end time	# of packets (millions)	GB transferred
WEST-01A	OC-3	Wed 9Aug00 9:56 PST	Thur 10Aug00 9:55 PST	817	483
WEST-01B	OC-3	Wed 9Aug00 9:56 PST	Wed 9Aug00 18:27 PST	326	108
WEST-02A	OC-3	Wed 9Aug00 9:56 PST	Wed 9Aug00 21:20 PST	853	397
WEST-02B	OC-3	Wed 9Aug00 9:56 PST	Thu 10Aug00 1:30 PST	853	298
WEST-03A	OC-3	Wed 9Aug00 9:56 PST	Wed 9Aug00 13:39 PST	284	169
WEST-03B	OC-3	Wed 9Aug00 9:56 PST	Wed 9Aug00 19:34 PST	568	292
WEST-04A	OC-3	Wed 9Aug00 9:56 PST	Thu 10Aug00 6:48 PST	568	175
WEST-04B	OC-3	Wed 9Aug00 9:56 PST	Thu 10Aug00 5:06 PST	853	468
WEST-05A	OC-3	Wed 9Aug00 9:56 PST	Wed 9Aug00 19:57 PST	568	193
WEST-05B	OC-3	Wed 9Aug00 9:56 PST	Wed 9Aug00 23:18 PST	853	461
WEST-06A	OC-3	Wed 9Aug00 9:56 PST	Wed 9Aug00 17:55 PST	284	121
WEST-06B	OC-3	Wed 9Aug00 9:56 PST	Thu 10Aug00 5:03PST	568	244
WEST-07A	OC-3	Wed 9Aug00 9:56 PST	Thu 10Aug00 9:55 PST	451	184
EAST1-08A	OC-12	Wed 5Sep01 8:00 EST	Wed 5Sep01 19:02 EST	110	36.0
EAST1-08B	OC-12	Wed 5Sep01 8:00 EST	Wed 5Sep01 18:06 EST	1680	623
EAST1-09A	OC-12	Wed 5Sep01 8:00 EST	Wed 5Sep01 14:17 EST	1120	645
EAST1-09B	OC-12	Wed 5Sep01 8:00 EST	Fri 7Sep01 17:47 EST	1680	509
EAST1-10A	OC-12	Wed 5Sep01 8:00 EST	Thu 6Sep01 10:05 EST	1680	972
EAST1-10B	OC-12	Wed 5Sep01 8:00 EST	Wed 5Sep01 15:24 EST	1680	626
EAST2-11A	OC-12	Wed 5Sep01 8:00 EST	Thu 6Sep01 8:00 EST	1340	313
EAST2-11B	OC-12	Wed 5Sep01 8:00 EST	Thu 6Sep01 01:51	1340	1110
EAST2-12A	OC-12	Wed 5Sep01 8:00 EST	Thu 6Sep01 00:07	1340	690
EAST2-12B	OC-12	Wed 5Sep01 8:00 EST	Wed 5Sep01 21:03	1120	459
EAST2-13A	OC-12	Wed 5Sep01 8:00 EST	Thu 6Sep01 4:54 EST	1680	1150
WEST-14A	OC-12	Wed 5Sep01 5:00 PST	Thu 6Sep01 21:02 PST	839	673
WEST-14B	OC-12	Wed 5Sep01 5:03 PST	Mon 10Sep01 00:06 PST	1360	240
WEST-15A	OC-12	Wed 5Sep01 5:00 PST	Wed 5Sep01 19:13 PST	206	46.9
WEST-15B	OC-12	Wed 5Sep01 5:00 PST	Wed 5Sep01 18:07	201	175
WEST-16A	OC-12	Wed 5Sep01 5:00 PST	Wed 5Sep01 13:35 PST	1680	905
EAST1-17A	OC-48	Fri 19Apr02 13:00 EST	Fri 19Apr02 14:00 EST	908	335
EAST1-17B	OC-48	Fri 19Apr02 13:00 EST	Fri 19Apr02 14:00 EST	667	380
EAST1-18A	OC-48	Fri 19Apr02 13:00 EST	Fri 19Apr02 14:00 EST	590	253
EAST1-18B	OC-48	Fri 19Apr02 13:00 EST	Fri 19Apr02 14:00 EST	996	615
WEST-19A	OC-48	Fri 19Apr02 10:00 PST	Fri 19Apr02 11:00 PST	1142	533
WEST-19B	OC-48	Fri 19Apr02 10:00 PST	Fri 19Apr02 11:00 PST	1319	723
WEST-20A	OC-48	Fri 19Apr02 10:00 PST	Fri 19Apr02 11:00 PST	341	170
WEST-20B	OC-48	Fri 19Apr02 10:00 PST	Fri 19Apr02 11:00 PST	621	280

Table 2.3: Traces collected

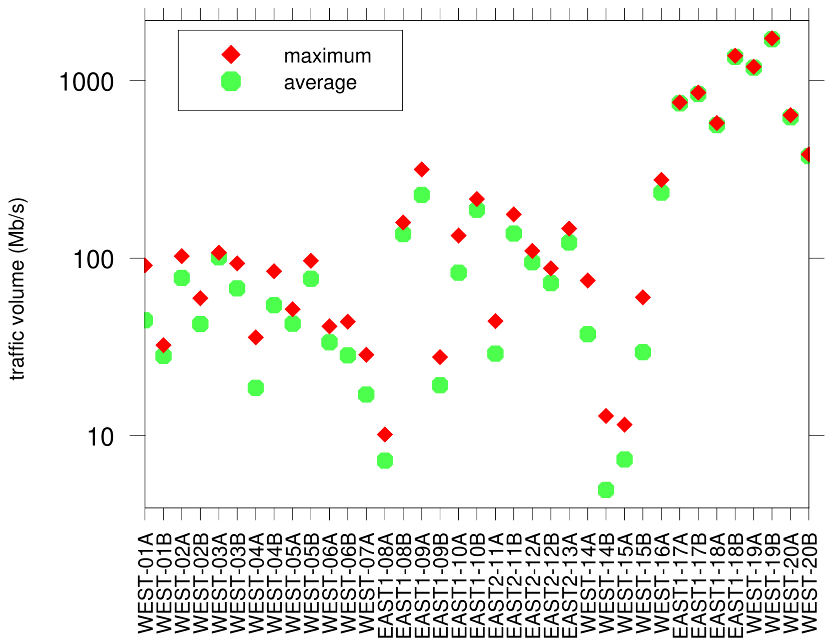


Figure 2.5: Traffic volume in Mb/s

volume observed on each link. The peak traffic volume represents the maximum amount of traffic seen over any five minute interval during the trace. The average traffic volume ranges from 5 Mb/s to 1.72 Gb/s, while the peak traffic volume ranges from 10 Mb/s to 1.73 Gb/s. For many of the links, the peak arrival rate is not much greater than the average arrival rate. This would seem to suggest that these links do not exhibit the same daily variations as observed in [108], which found that traffic volume during the day was three times greater than the traffic volume at night. However, not all of our traces contain a full 24 hours of data, and we therefore cannot always observe a daily trend. For the traces that do contain a full 24 hours of data, the peak arrival rate is between 1.5 and 2 times the average arrival rate.

The monitored links have a capacity of either 155 Mb/s, 622 Mb/s, or 2.5 Gb/s.

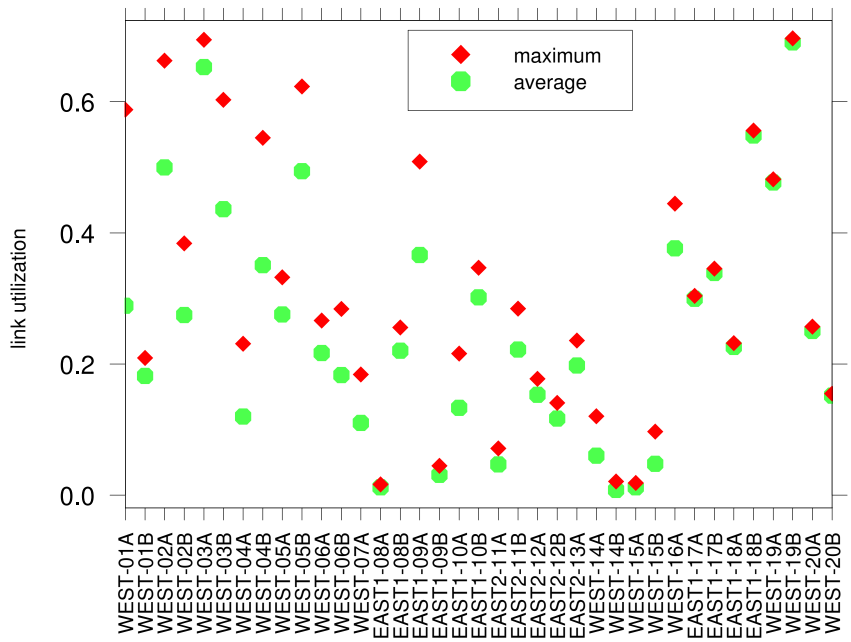


Figure 2.6: Link utilization

In many cases, the average traffic volume is much less than the link capacity. Figure 2.6 plots the average link utilization for each of the monitored links. Only three links, WEST-03A and EAST-19A, and WEST-19B have an average utilization greater than 50%. Only nine links have a peak utilization greater than 50%. The highest peak utilization is 70%, and it was observed on WEST-19B.

The low link utilization is an intentional design choice which was made so that little queuing delay occurs in the network. Later in this section we present measurements of delay through the backbone that demonstrate that this practice is successful.

Figure 2.7 shows the peak and average traffic volume in terms of thousands of packets/sec. This traffic metric represents the amount of load placed on network routers and other equipment that must perform a fixed operation (e.g. a destination

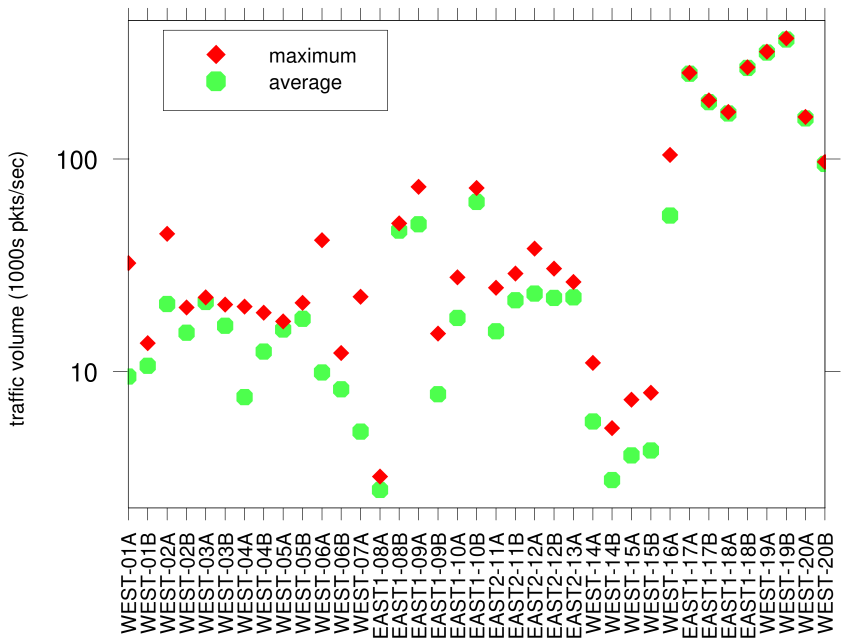


Figure 2.7: Traffic volume in packets/sec

address lookup) for each packet. In fact, it is a useful metric to evaluate the design of the IPMON measurement entities. As described in Section 2.3.2, the OC-3 and OC-12 measurement entities were designed to be able to record data at an average rate of 469,000 packets/s and the OC-48 were designed to collect data at an average rate of 1.4 Mpackets/s. The figure shows that the actual packet rate is much lower than the maximum which can be supported by the IPMON measurement entities.

Next we investigate the composition of the observed traffic. Figure 2.8 shows the percentage of TCP, UDP, ICMP, IPv6, and other traffic on each link. For all but five traces, TCP accounts for over 90% of the traffic. For those five traces, UDP is the source of the majority of the remaining traffic except for EAST2-18A for which 20% of the traffic is ICMP traffic. The source of the UDP and ICMP traffic is described

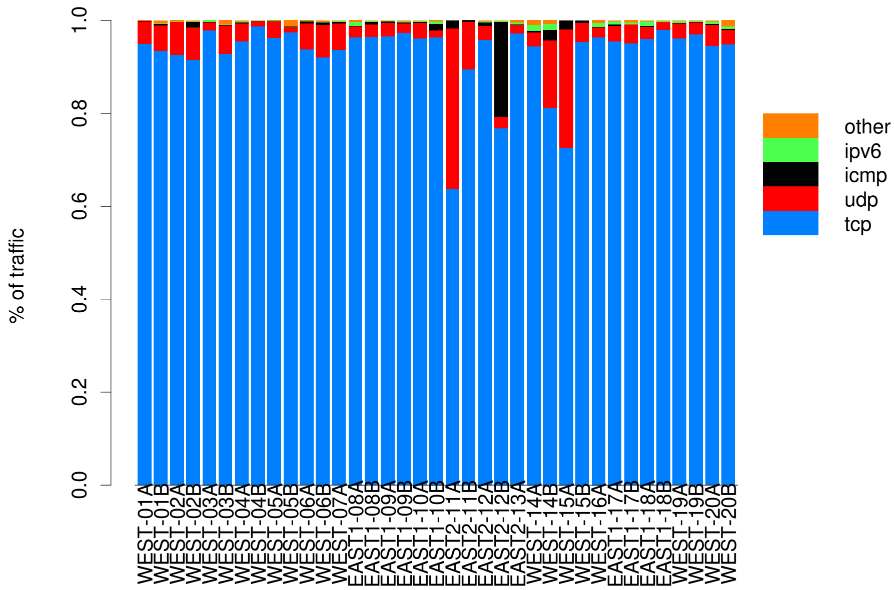


Figure 2.8: Traffic composition by IP protocol

in the discussion on the application mix.

Figure 2.9 shows the traffic composition by application. We consider six types of applications: web, distributed file sharing, streaming media, email, ftp, and other traffic on each of the links. We identify the applications using the TCP and UDP port numbers. Applications such as web, email, and ftp use official well-known TCP port numbers. The ports used by distributed file sharing and streaming media applications, however, are not officially assigned and depend on the implementation decisions made by the application developers. To identify these applications, we use the port numbers used by several popular implementations. For distributed file sharing applications this includes Napster, Kazaa, and Morpheus while streaming media includes Microsoft Media Player, RealAudio, and RealVideo. We also include traffic with the well-know

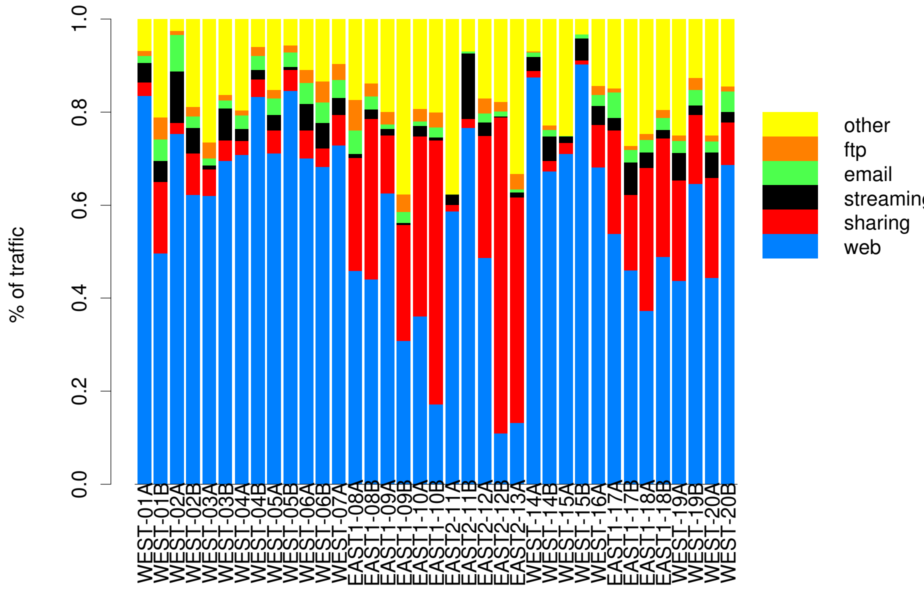


Figure 2.9: Traffic composition by application

application	port numbers
web	http: 80, 8080 https: 443
distributed file sharing	Napster: 4329, 4444, 5555, 6666, 6688, 6697, 6699, 7777,8888 Gnutella: 6346 Scour: 8311 iMesh: 4000-5000 KaZaA: 1214
streaming media	rtp/rtsp: 554 RealAudio: 6970-7170 Microsoft Media Player/NetShow: 1755 Shoutcast: 8000, 8001, 8600, 8700, 8800, 8888 CU-SeeMe: 7648-7652, 24032 Liquid Audio: 18888, 18889
email	smtp: 25 POP: 109,100 IMAP: 143,220
ftp	ftp: 20,21

Table 2.4: TCP/UDP port numbers for various applications

RTP port numbers in the streaming media category. The port number information was obtained from the CoralReef traffic analysis program [61] and is shown in Table 2.4. Traffic which has port numbers other than those listed in the table is categorized as *other*.

For all of the traces collected in August 2000, web traffic accounts for the majority of the traffic. However, for the traces collected in September 2001 and April 2002, we see that web is much less dominant on many links. For 13 of the 24 traces, web traffic accounts for less than 50% of the traffic on the link. The reason for this is the large volume of filesharing traffic on these links. This can be seen most dramatically on link EAST2-12B, where web accounts for only 11% of the total traffic, while filesharing traffic accounts for 68% of the traffic volume. This link is connected to an access router which is in turn connected to a customer with a large population of DSL home users. An interesting question is: does the file sharing traffic account for the large amount of UDP traffic seen on several of the September 2001 traces? In these measurements, most of the filesharing traffic uses TCP. The UDP traffic corresponds to the other applications. We have not been able to identify the specific application which generated this traffic.

Another point of interest is the large amount of streaming media traffic on link EAST2-17B which is connected to an access router which is in turn connected to a large content distribution network (CDN). Apparently, the CDN is hosting a large volume of streaming media which accounts for 14% of the traffic on this link.

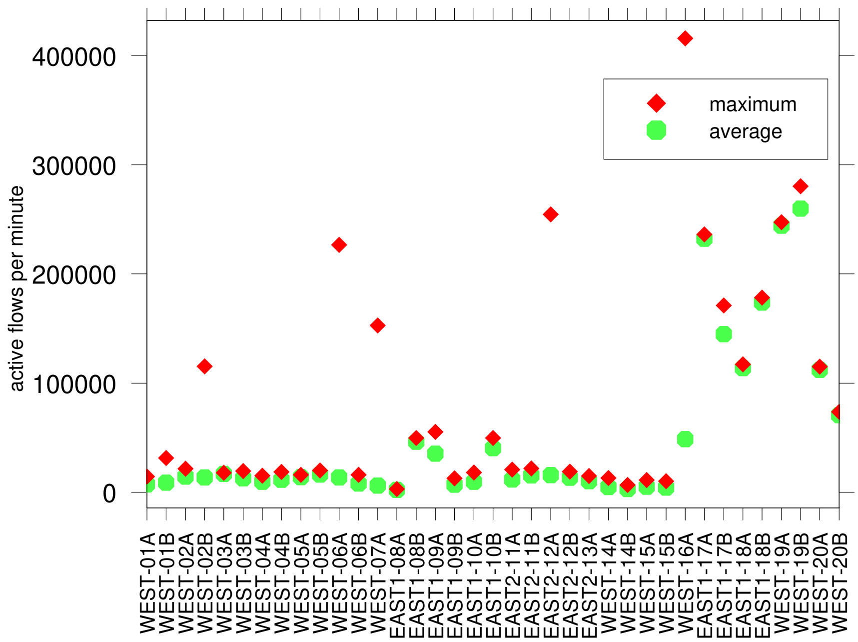


Figure 2.10: Active flows per minute

2.4.4 Flow Characteristics

In this section we analyze the characteristics of the individual user flows observed in each trace. As described in Section 2.3.4, a flow is defined as a sequence of packets all of which have the same source IP address, destination IP address, and IP protocol field. For TCP and UDP traffic, the packets must also have the same source and destination port. If there is an idle time greater than one minute between consecutive packets, we consider the new packet to represent a new flow.

First we investigate the number of active flows. For each trace, we divide the trace into one minute intervals and compute the number of flows which actively transmitted data during the one minute interval. Figure 2.10 shows the average and maximum number of active flows for each trace. In the Aug. 2000 and Sept. 2001 traces, the average and maximum number of flows per minute is typically between 10,000 and 50,000. There are a few exceptions. The maximum number of flows per second for trace EAST2-12A is 254,000, while the average is only 16,000. One may expect this to be related to the large volume of ICMP traffic observed on this link. However, this is not the case. Traces from EAST2-12A, as well as WEST-02B, WEST-05A and WEST-07A, WEST-16A observe a malicious user behavior known as a TCP port scan. In a TCP port scan, a malicious user transmits either a large number of TCP SYN packets or a large number of TCP FIN packets to different hosts in an attempt to detect systems which have bugs that can be exploited to gain remote access. This results in a large number of individual user flows since each packet has a different port number or destination address, and does increase the maximum traffic rate in terms of packets per second as seen from Figure 2.7. However, it does not significantly increase the traffic volume in terms of bits per second. Consider the case of EAST2-12A which has a maximum of 254,000 flows in a one minute interval. The majority of these flows are from the TCP port scan, but each flow only transmits a single 40 byte packet. This represents an increase in traffic volume of only 1.4 Mb/s, or about 1.5% of the average traffic volume observed on this link.

The Apr. 2002 traces were collected on high bandwidth OC-48 links, and carry a significantly larger number of flows than seen in the earlier measurements. The average number of flows seen on these links ranges from 70,000 to 260,000. The maximum number of flows is not significantly larger than the average since each of these traces lasted only for one hour.

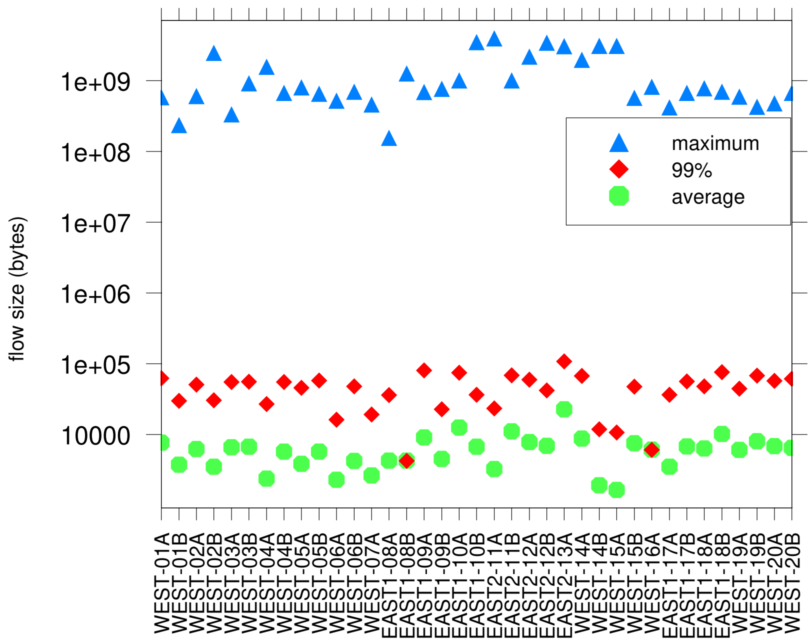


Figure 2.11: Flow size in bytes

Figure 2.11 shows the average flow size (in terms of bytes), the maximum flow size, and the 99^{th} percentile of the flow size distribution. On all but four links, the average flow size is less than 10,000 bytes. One link, EAST2-13A, has a significantly larger average flow size (23,000 bytes) than most other traces. The large flows on this link are generated by the distributed file sharing applications. However, other links with large volumes of file sharing traffic (e.g. EAST-12B) have similar flow size distribution to links which carry primarily web traffic. While the file sharing traffic has the potential to increase the average flow size in the network, it is not strictly the case that this occurs.

Figure 2.11 also demonstrates that there are a small number of extremely large flows in the network. All of the links have at least one flow which transfers at least

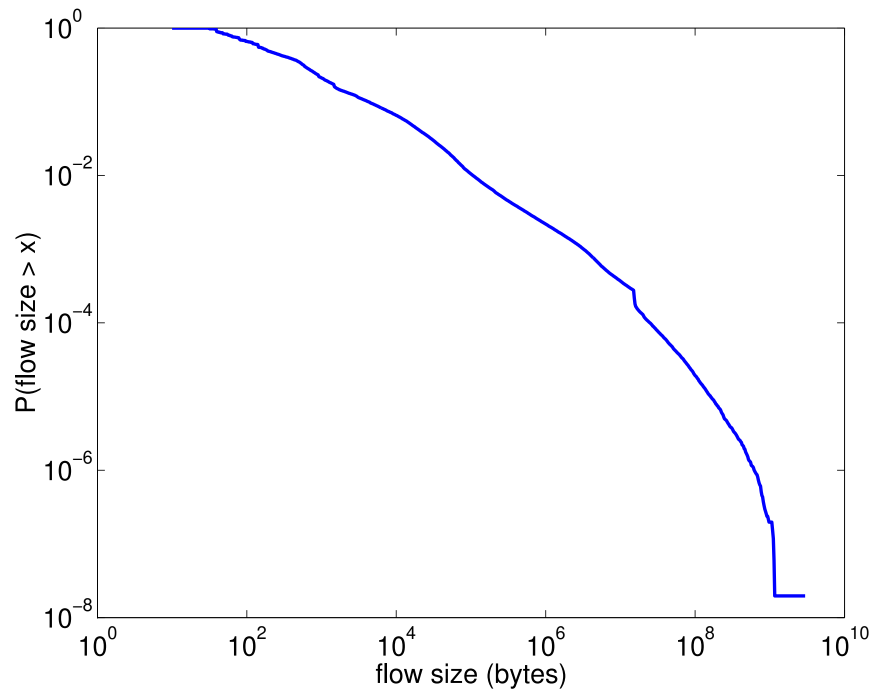


Figure 2.12: Flow size distribution

150 MB of data. 25 of the links have flows which transfer more than 1 GB of data. These flows, however, represent a very small fraction of the total flows observed on each link. To demonstrate this, we plot the complementary distribution of flow sizes from EAST2-13A (the link with the highest average flow size) in Figure 2.12. On this link, only 1% of the flows transfer more than 108 KB of data. Only 0.04% of the flows transfer more than 10 MB of data. Rather than repeat Figure 2.12 for all traces, we instead show the 99^{th} percentile of the flow size distribution for all links in Figure 2.11. For most links, 99% of the flows transfer less than 100 KB of data.

This observation is consistent with results found in [90], [112], and [24] which found that for many types of network traffic (LAN, WAN, web, etc.), there is a non-negligible probability of observing extremely large flows. Distributions which

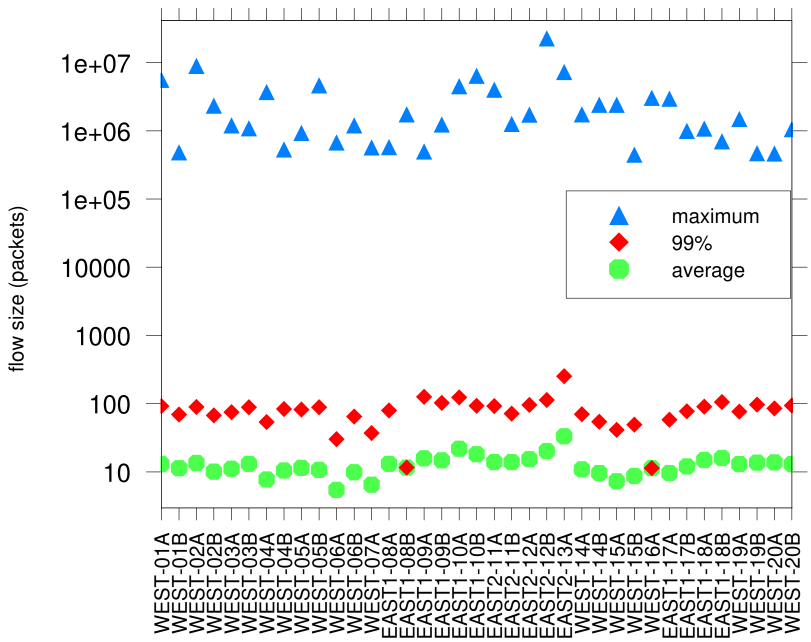


Figure 2.13: Flow size in packets

have this property are known as *heavy-tailed distributions.* We perform a statistical analysis of the flow size distributions in Chapter 3 and find that for most of the traces, the flow size distribution is heavy-tailed.

Figure 2.13 presents a similar flow size analysis, but in terms of packets rather than bytes. We see from this figure that the average flow size in packets is quite small; typically between 10 - 15. Similar to the results seen in Figure 2.11, there do exist a small number of very large flows which transmit over 1 million packets. These flows, however, represent a small fraction of the total number of flows.

Figure 2.14 shows information about the duration of individual user connections. For all but one trace, the average connection duration is less than 10 sec, and 99% of the flows last less than 100 sec. The flows observed in trace EAST2-13A, the trace

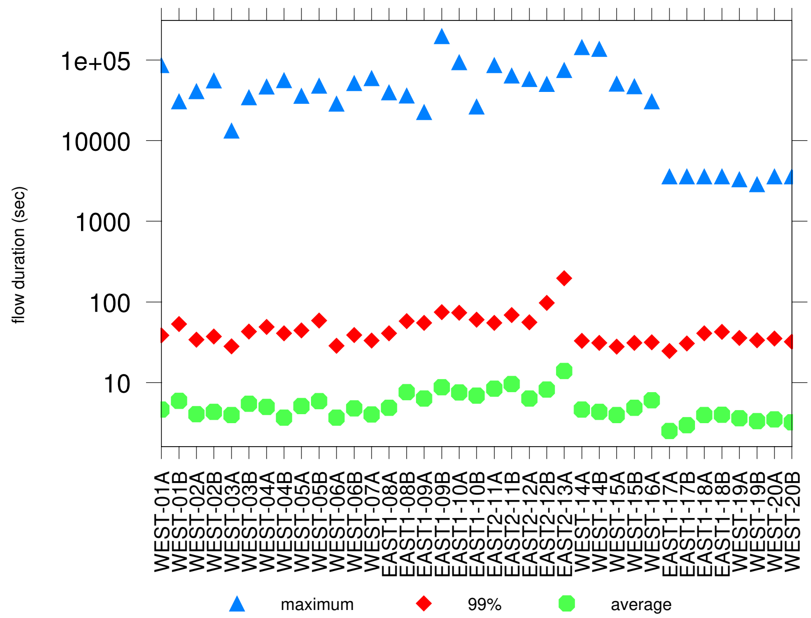


Figure 2.14: Flow duration

with the large average flow size, have an average connection duration of 14 sec. The 99^{th} percentile of the flow duration distribution for this trace is 196 sec, twice as large as the 99^{th} percentile of any other trace. In the traces collected in Aug. 2000 and Sept. 2001, a small number of flows can last over 3 hours (10800 seconds). Flows which last more than 2 hours are mostly streaming media applications (RTP and RealAudio) and large web connections. These web connections could be large file downloads or they could be streaming media applications which are required to use http in order to pass through a firewall. However, since we do not collect any of the user data contained in the packet, we are unable to determine what type of data is transferred in these large web connections.

Next we investigate the traffic rate for the user connections. Figure 2.15 plots the

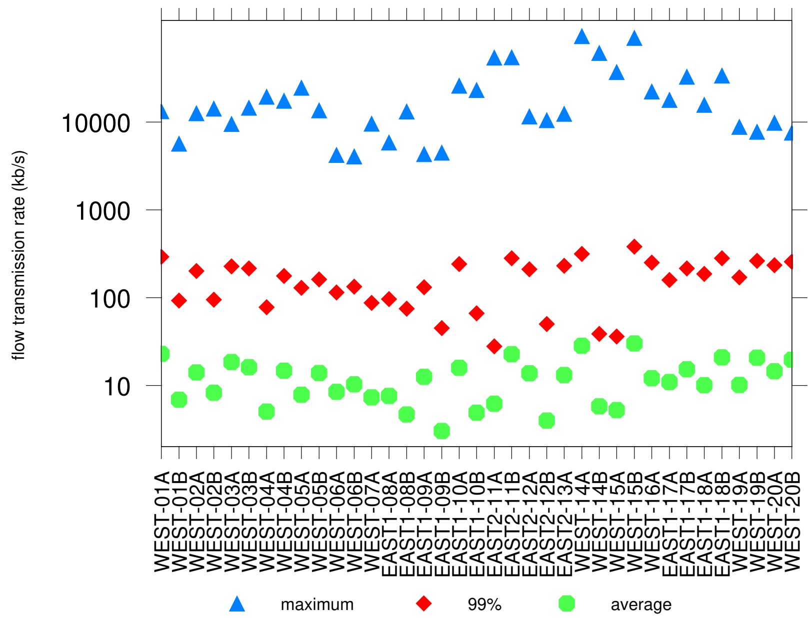


Figure 2.15: Flow rates

average, 99^{th} percentile, and maximum rates of individual user flows for each of the traces. When plotting the flow rates, we do not consider any flows which last less than 1 second. The average rate of an individual user flow is less than 30 kb/s for all of the traces. This low average rate is not the result of limited bandwidth in the network (even a slow speed modem link has a bandwidth of 56 kb/s). Instead, these flows are limited in their transmission rate by the TCP slow start mechanisms. A TCP session begins by the sender transmitting a SYN packet to the destination in order to establish the connection. After the remote system responds with a SYN-ACK, the sender will transmit a single data packet. After the sender receives an acknowledgement (ACK) from the receiver that the first packet was received correctly, the sender will transmit two more data packets. This procedure repeats, and the number of packets transmitted doubles after each set of packets is acknowledged (assuming no loss). From Figure 2.13, an average connection contains 10 - 15 packets (including the SYN packet). If the round-trip-time (rtt) between when a packet is transmitted and a when a packet is received is 100 ms, it will take 500 ms to transmit all of the data in the connection. With an average flow size of 10,000 bytes (from Figure 2.11), the rate of such a connection will be 20 kb/s.

Figure 2.15 also shows that 99% of flows have a rate less than 300 kb/s. This rate is much smaller than the bandwidth of the links from which the measurements were collected. However, a small number of flows can reach rates of 30 - 50 Mb/s. These high rate flows, however, are very brief in duration. They typically last only between 1 - 5 seconds. Flows which last more than one minute never exceed 1 Mb/s. On two links, WEST-14B and WEST-15B, flows with rates between 90 - 100 Mb/s have been observed. These two links happen to be connected to a large Content Distribution Network (CDN) customer. The very high bandwidth flows correspond to connections directly between servers operated by the CDN customer. Similar to the high bandwidth flows observed on other links, these connections typically last no more than a few seconds.

Figure 2.16 plots the median round-trip-time (rtt) for flows observed in each trace. We estimate the rtt using the procedure described in Section 2.3.4. For most traces, the median rtt is between 80 ms and 300 ms. Four traces have median rtt's less than 80 ms. In particular, trace EAST2-11B has a median rtt of only 12 ms, WEST-14A has a median rtt of 31 ms, and WEST-15B has a median rtt of 58 ms. The low rtt's

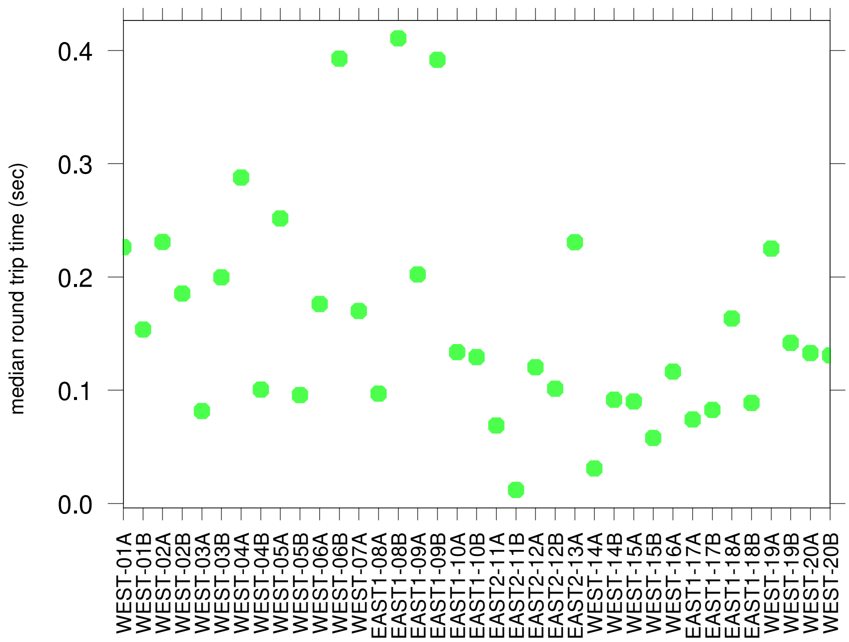


Figure 2.16: Flow round-trip times

are the result of the particular customer which is connected to these links. All three of these links are connected to a large Content Distribution Network (CDN). The goal of the CDN is to direct web requests to a server which is not heavily loaded and which has a short path between the user and the server. The low rtt's are evidence that the CDN is achieving its goal of directing requests to servers which have a relatively short path to the end user. Figure 2.16 also shows three traces with a median rtt of nearly 400 ms. Two of these links, EAST-08B and EAST-09B are connected to European ISPs, and are therefore expected to have large rtt's. The remaining link, WEST-06B is connected to a large domestic Tier-2 ISP. We do not have a satisfactory explanation for the large rtt's observed on this link.

Finally, we investigate the end-to-end loss experienced by TCP flows. As described

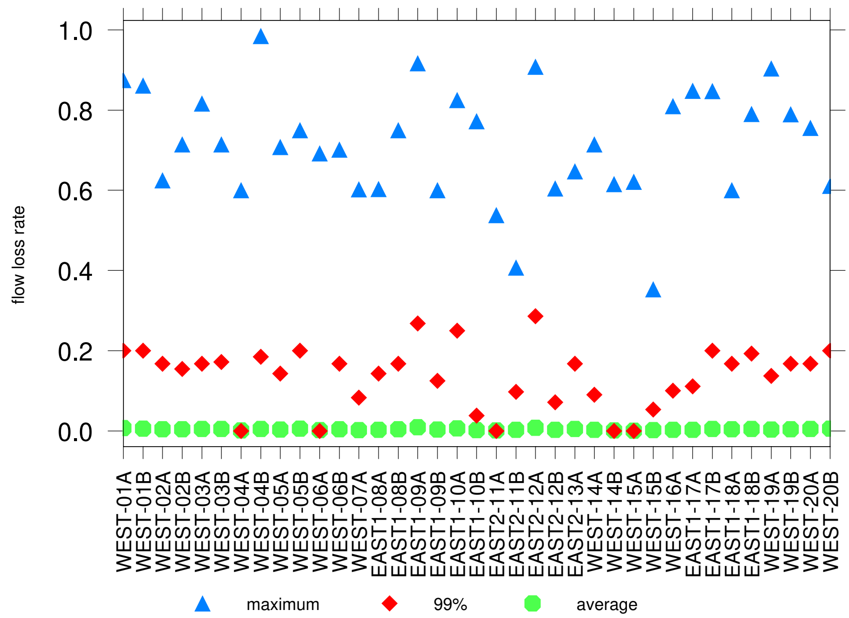


Figure 2.17: TCP loss rates

in Section 2.3.4, the end-to-end loss rate for a flow can be inferred by inspecting the TCP sequence number of each packet in a flow. This number therefore includes losses which occur outside of the Sprint network. However, since we monitor a link which is in the middle of the end-to-end path, there are some losses we cannot detect (e.g. if the first SYN packet is lost before it reaches the monitored link). The loss rates are therefore a lower bound on the actual loss rates seen by the TCP flows.

Figure 2.17 plots the average end-to-end loss rate for a TCP flow, the maximum loss rate, and the 99^{th} percentile of the loss rate distribution for each trace. In one trace, the average loss rate for a flow is less than 1.5%, and four of the traces have an average loss rate between 0.5% and 1%. For the remainder, the average loss rate is less than 0.5%. While most flows experience very little or no loss, some flows do have a large percentage of lost packets. In most traces, 1% of the flows have a loss rate greater than 10%.

2.4.5 Delay Measurements

The Sprint IP backbone is designed so that packets experience very little queuing delay in the network. As described in the Introduction, this is accomplished by installing enough capacity in the network so that link utilization does not exceed some threshold. While the exact value of this threshold is proprietary, we see from Figure 2.6 that link utilization on the monitored links rarely exceeds 50%.

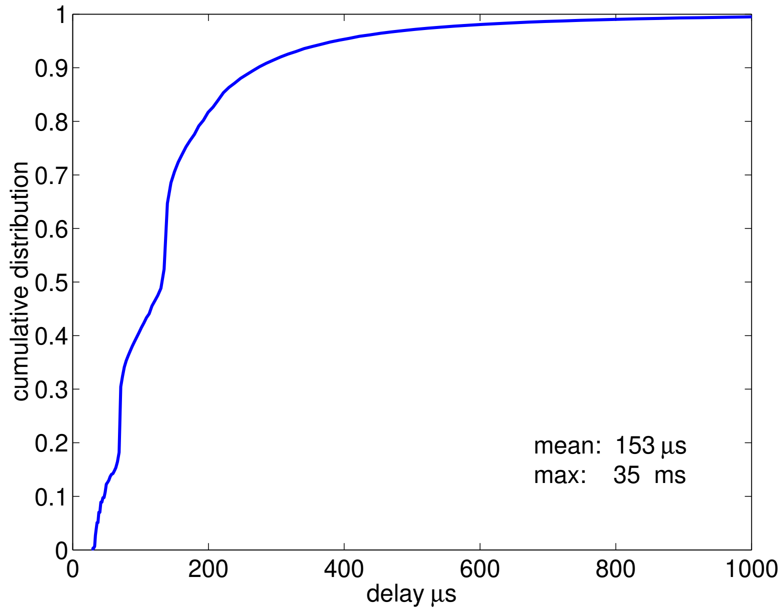
In this section we investigate if this design approach is successful at maintaining low queuing delay in the Sprint backbone. Using traffic measurements collected by the IPMON system, we can measure the delay experienced between two monitored links in the network using the procedure described in Section 2.3.4. We first analyze the delay experienced through a single router in the network, and then analyze the delay experienced between the East and West coast POPs.

Delay Through a Single Router

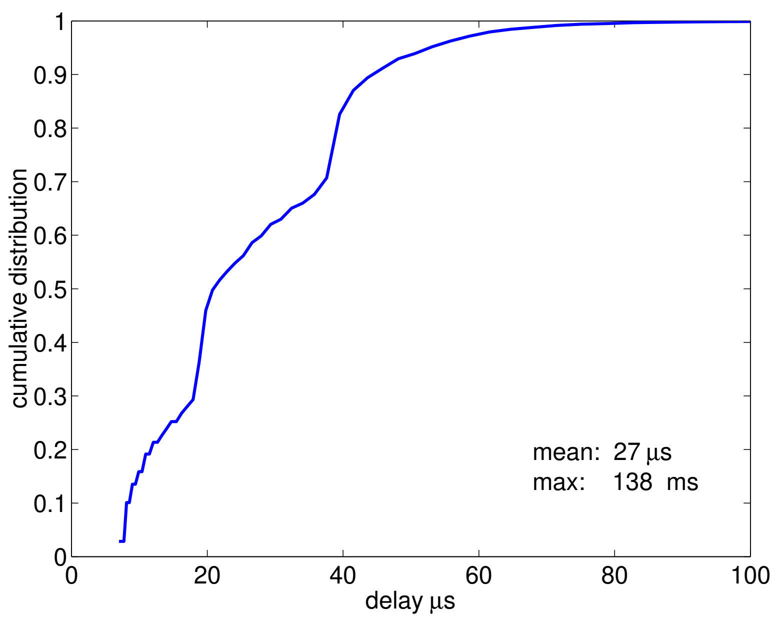
We begin by analyzing the delay experienced through a single router in the Sprint IP backbone. For the measurements collected on Aug. 9, 2000, the links WEST-05B and WEST-01A were connected to the same router. WEST-05B was an input to the router and WEST-01A was an output of the router. Using the traces we can measure the delay for all packets which arrived to the router on WEST-01A and departed on WEST-05B. Similarly, for the measurements collected on Sept. 5, 2001 the link EAST1-08B was the input to one router, and EAST1-09A was an output of the same router.

Figure 2.18 plots the cumulative distribution of the packet delays observed between the two pairs of links. For the Aug. 2000 measurements, the average packet delay through a single router is only 153 μs, and 99% of the packets have a delay less than 804 μs. For the Sept. 2001 measurements, the average packet delay is only 27 μs, and 99% of the packets have a delay less than 72 μs. One reason for this decrease is that the WEST-05B link has an average utilization of 49% while the EAST1-09B link has an average utilization of 37%. The lower utilization should result in somewhat smaller queuing delays. However, there is a second factor which is more significant. The Aug. 2000 measurements were collected on 155 Mb/s OC-3 links, while the Sept. 2001 measurements were collected on newer 622 Mb/s OC-12 links. The service time of a packet on an OC-12 links should be four times smaller than that of an OC-3 link. From the figure we see that this is indeed the case. The minimum delay from the Aug. 2000 measurements is 28 μs while the minimum delay for the Sept. 2001 measurements is only 7 μs. This minimum delay represents the amount of time it takes the router to perform the route lookup, switch the packet to the output interface, and transmit the packet on the outgoing link. The minimum processing time for an OC-12 link is, as expected, four times smaller than that of an OC-3 link.

However, in both sets of measurements, there are a small number of packets which have very long delays (over 100 ms in the case of the Sept. 2001 measurements). To

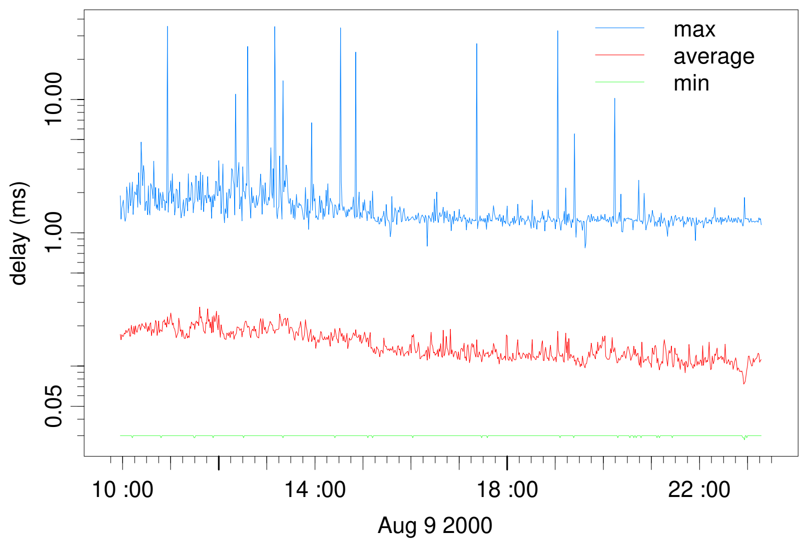


(a) WEST-05B to WEST-01A

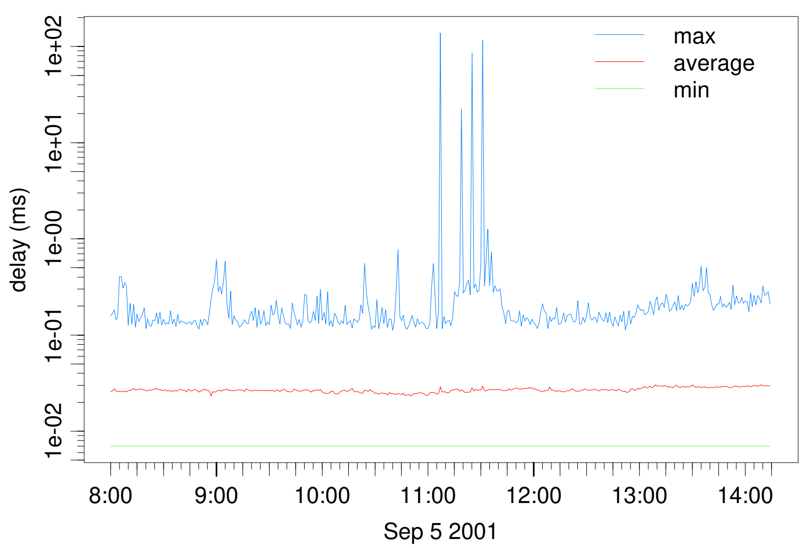


(b) EAST1-08B to EAST1-09A

Figure 2.18: Delay through a single router



(a) WEST-01A to WEST-05B

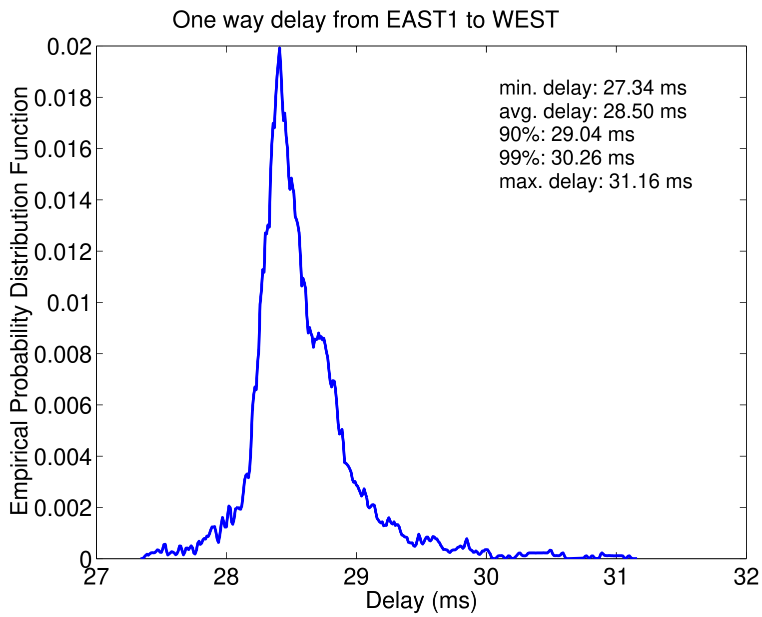


(b) EAST1-08B to EAST1-09A

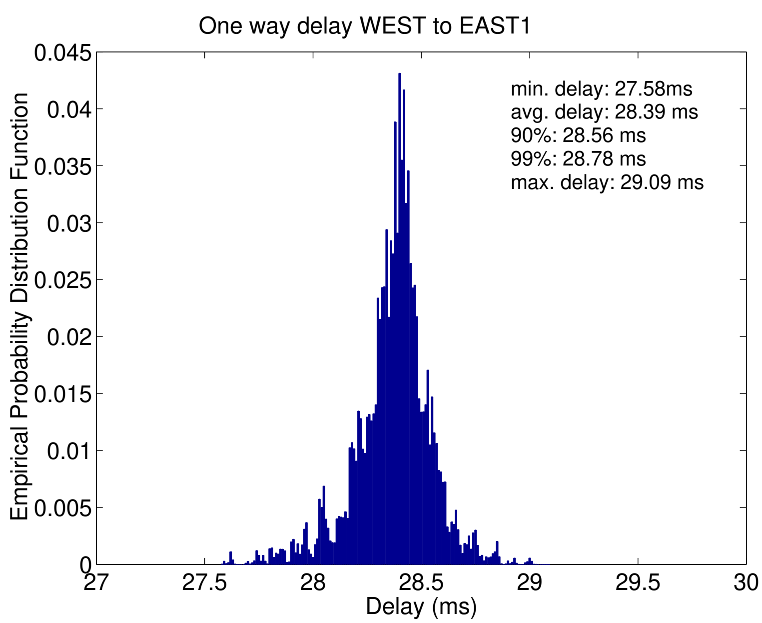
Figure 2.19: Single router delay vs. time

investigate these long delays we plot the minimum, average, and maximum delay for each one minute interval of the measurements. This data is shown in Figure 2.19. For the Aug. 2000 measurements, the minimum delay over each one minute interval is approximately 28 μs. The maximum delay is typically between 1 ms and 2 ms, but there are occasional spikes which reach up to 35 ms. A similar behavior is seen in the Sept. 2001 measurements, with the minimum consistently at 7 μs, and the maximum typically between 100 μs and 1 ms. Between 11:00 and 12:00, however the Sept. 2001 measurements also exhibit quite large spikes.

There are two possible causes for the large spikes seen in Figure 2.19. One possible cause is that a burst of packets arrived at the queue for the monitored output link and caused massive delays. A second possible cause is that the router was busy updating routing tables, processing SNMP requests, or performing some other action which prevented it from forwarding packets for a period of time. If the delays were caused by a burst of packets, we would expect to observe this burst of packets on the output link. However, the measurements do not indicate that such a burst is present. In fact, during the time at which packets were waiting in the router, the output link was idle. This indicates the long delays are caused by a period of time in which the router is unable to forward packets. A detailed analysis of the different possible causes of this idle time (e.g. route lookups, head-of-line blocking) can be found in [86]. In that study we find that these idle periods affect less than 1% of the total traffic.



(a) EAST1-18A to WEST-19A



(b) WEST-19B to EAST1-18B

Figure 2.20: Delay distribution between EAST1 and WEST POPs

Delay Between East and West Coast POPs

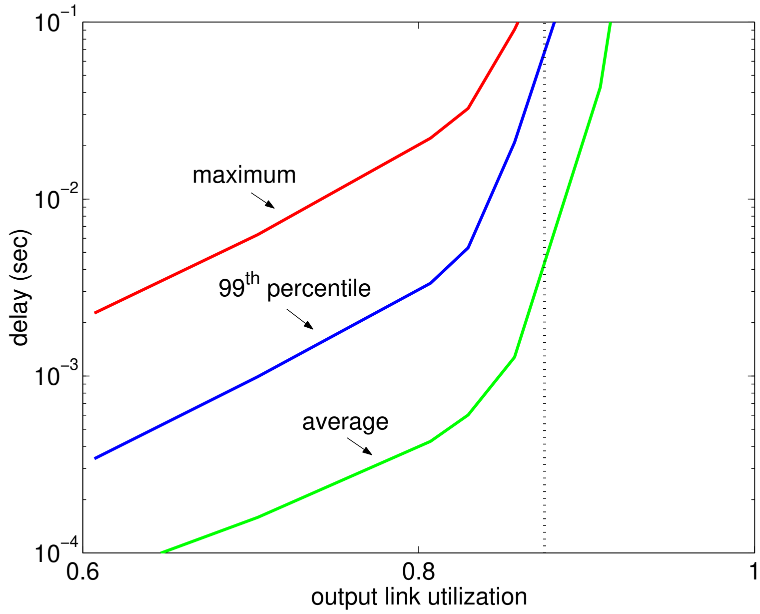
Next we investigate the delay between the EAST1 POP and the WEST POP. Figure 2.20 shows the distribution of packet delay from EAST1-18A to WEST-19A and from WEST-19B to EAST1-18B. In both cases, the minimum delay is approximately 27.5 ms. This corresponds to the propagation delay and the minimum processing time required at each router along the path between the two sites. From EAST1 to WEST, the average delay is 1.16 ms greater than the minimum delay and the maximum delay is only 3.82 ms greater than the minimum. These values represent the average and maximum queuing delay that is experienced by traffic between these two sites. In the opposite direction, from WEST to EAST1, the average queuing delay is 0.81 ms, while the maximum queuing delay is only 1.51 ms.

2.4.6 Delay Simulations

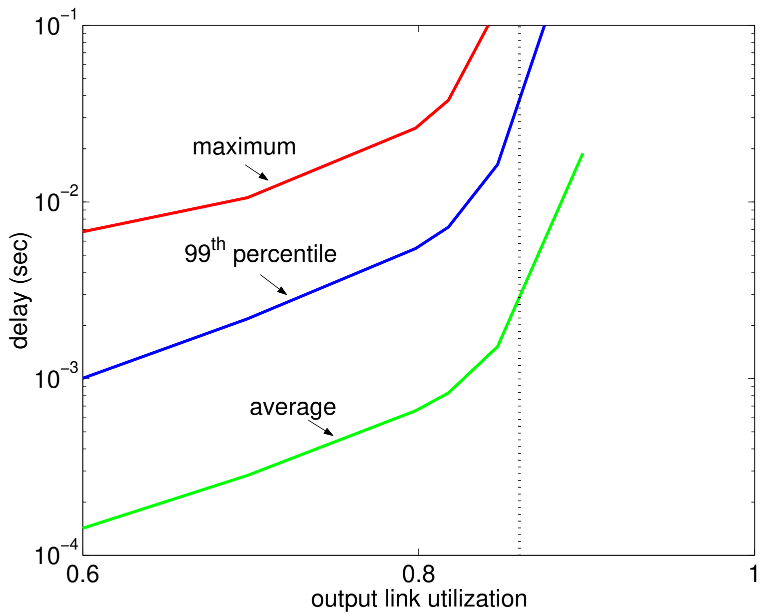
In the previous section we observed that little queuing delay occurs in the Sprint IP backbone. This is not unexpected as the network is designed to operate a low link utilization in order to minimize queuing delay. To evaluate the bandwidth provisioning problem, however, we must know at what level of link utilization queuing delays begin to increase. Since we are unable to observe this directly from the network due to the low link utilizations, we investigate this question through simulation.

We developed a queuing simulator which reads a packet trace and simulates injecting the traffic into an infinite buffer queue served by a constant bit rate server of capacity C. Using the simulator, we can determine the delays experienced by each packet in the trace if the link capacity were reduced (or conversely, if the link utilization were increased).

There are three factors that would cause the delay in the simulator to be different than the delay that the traffic would see in the actual network. The first source of error is that a router does not implement an ideal FIFO queue. As we saw in Figure 2.19, some packets experience very long delays due to router pathologies. However, these router behaviors affect a very small fraction of the packets. Furthermore, the intent is not to model the behavior of one particular model of router, but rather to understand the characteristics of backbone traffic and how they affect queuing delay. The second source of error is the simulator emulates an infinite buffer queue, while the buffers in the actual network are finite. Typical buffer sizes in the actual network correspond to between 250 ms and 1 second of queuing delay. In the simulation results we present, the maximum delay observed in the simulator is typically less than 100 ms for the range of link utilizations in which we are interested. In such cases, no loss would be experienced in the actual network. The final source of error is that we do not consider the feedback mechanism of TCP. If we simulate a network in which there is a large amount of loss or large delays, the TCP congestion control mechanism would cause the sources to slow their transmission. We do not account for this behavior in our simulator. However, as we shall see, over the range of link utilizations in which we are interested, only a small number of packets experience delays greater than several milliseconds and no packets experience loss. In such a case the TCP feedback mechanism would have minimal impact.



(a) WEST-04B



(b) WEST-02B

Figure 2.21: Delays observed in simulation

Figure 2.21 shows the average, maximum, and 99^{th} percentile of the delays observed in simulations performed with traces from links WEST-04B and WEST-02B. Multiple simulations were performed for each trace, with link utilizations ranging from 0.6 to 0.95. The behavior for link utilization less than 0.6 continues the trend shown in the figure. The dotted line shows the point at which the maximum delay exceeds 250 ms. For link utilizations to the right of this line, the delay simulations may exhibit the errors described above.

The figure demonstrates that below utilization of 0.8, only 1% of the traffic experiences queuing delays greater than a few milliseconds. Furthermore, the maximum delay observed at 0.8 utilization is just slightly greater than 20 ms. However, between utilization of 0.8 and 0.9, the delays begin to rapidly increase. Simulations performed with traffic from the other links exhibit the same type of knee behavior, with the knee occuring between 0.8 and 0.95 utilization. In particular, for traffic measured on the 2.5 Gb/s OC-48 links, the knee occurs above 0.90 utilization.

This knee behavior is of particular importance when deciding between bandwidth provisioning and traffic differentiation. Traffic differentiation is only needed if links are operated at utilizations above the knee. For the traffic measurements studied, differentiation would allow a network provider to operate safely between 0.8 to 1.0 utilization. Without differentiation, links utilization could still reach 0.8 to 0.95 depending on the particular traffic characteristics, and still satisfy most reasonable delay requirements.

2.5 Summary

In this chapter, we used packet-level traffic measurements collected by the IPMON measurement facility to study the characteristics of traffic in the Sprint IP backbone network. From this study , we can make two observations.

First, backbone IP traffic is aggregated from a large number of users (between 10,000 and 300,000 per minute), each of whom generates data at a rate much less than the total capacity of a backbone network link. These users are typically limited in their data rate by the TCP slow start mechanism or by limited bandwidth in the access networks. This type of traffic is fundamentally different than traffic observed in prior measurement studies, as well as traffic used in many network simulation studies. Most research studies consider network traffic which is aggregated from a small number of users and in which a single user can dominate the characteristics of the traffic on a single link. As we will see in the next chapter, because of the high level of aggregation in backbone networks, many statistical properties of backbone traffic follow a Gaussian distribution. As a result, such traffic is easier to model than traffic typically considered in the networking literature.

Second, the Sprint IP network is designed to maintain relatively low link utilization. On the links we monitor, link utilization rarely exceeds 50%. As a result of this design practice, traffic experiences minimal queuing delay in the Sprint backbone. There are, however, an extremely small number packets which experience delays over 100 ms, but these delays are the result of router behaviors such as route update processing, rather than network congestion. Through simulation we are able to show that even if link utilization were increased to between 80% and 90%, very few packets would experience more than several milliseconds of queuing delay. As a result, bandwidth provisioning is an attractive solution to meeting delay requirements in backbone networks.

Chapter 3

Evaluating Backbone Queuing Delay

3.1 Introduction

In Chapter 2 we demonstrated, through simulation, that link utilization on high speed links can reach 80% to 90% before queuing delay begins to exceed several milliseconds. While this simulation approach can be used to provision a network, it requires collecting detailed packet-level traffic measurements and performing lengthy simulations. Provisioning and other network design problems are greatly simplified by using an analytic traffic model. In this chapter we develop a model which captures the observed traffic arrival characteristics of measured backbone traffic. We call this model *two-scale Fractional Brownian Motion* (two-scale FBM). Using an approach originally proposed by Norros [83], we derive an expression for the delay distribution of a queue fed by two-scale FBM. We also develop a procedure to compute end-to-end queuing delays using this model.

For presentation purposes, we use a one hour segment of traffic from link WEST-04B to derive and explain the model. We then use a set of 331 one-hour measurements collected in August 2000 and September 2001 to validate the model. This validation is performed by comparing the delays computed analytically using the model with the delays observed in simulation using the 331 traces.

3.1.1 Queuing Delay Analysis

The model only needs to capture the traffic characteristics which affect queuing delay. We therefore begin by reviewing the procedure used to compute the queuing delay distributions. This procedure was originally developed by Norros in [82] and applied to LAN traffic measurements in [83].

Consider an infinite buffer queue with a constant bit rate server of capacity C. Let $A[s,t]$ be the amount of traffic that arrives at the queue over the time interval $(s,t]$, and let $A_t = A[-t,0]$. The queue length at time 0 is

$$Q = \sup_{t \geq 0}(A_t - Ct)$$

The probability that the queue length exceeds some value x is then

$$P[Q > x] = P[\sup_{t \geq 0}(A_t - Ct) > x]$$

In this form, $P[Q > x]$ is difficult to evaluate, so we use the lower bound

$$P[\sup_{t\geq 0}(A_t - Ct) > x] \geq \sup_{t\geq 0} P[A_t > x + Ct] \tag{3.1}$$

This may seem to be a rather crude approximation, but it has been shown to be logarithmically accurate for large x [32].

This model considers a router in the network to behave as an ideal output-queued router. Actual network routers will behave somewhat differently depending on the particular architecture of the router (e.g. input queued, combined input-output queued, etc.). Many of these router architectures, however, are designed to emulate the behavior of ideal output-queued routers. In [86] we measured the delay through routes in the Sprint network and found that it is reasonable to consider network routers to be ideal output-queued routers.

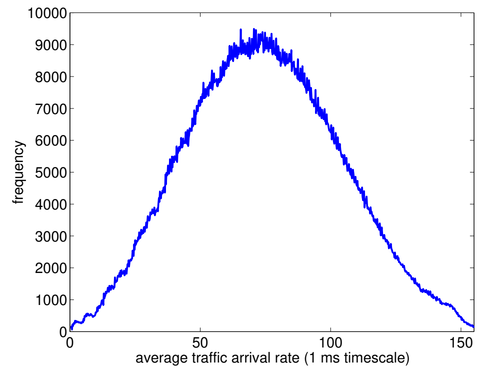
The queuing delay experienced by a packet of size b bits is the sum of the waiting time in the queue, $\frac{x}{C}$, and the service time of the packet, $\frac{b}{C}$. The distribution of the waiting time, W, is found directly from the queue length distribution $P(W > d) = \sup_{t\geq 0} P(A_t > C(d+t))$. The service time distribution is determined by the packet size distribution of the arrival traffic. Rather than model the complex packet size distributions which have been observed [43], we note that the transmission time of a maximum size packet is 80 μs on an OC-3 link (one of the lowest speed backbone links). Since this is much smaller than the delay bounds required by network customers we ignore the service time and consider the queuing delay to be equal to the waiting time.

3.2 Traffic Arrival Characteristics

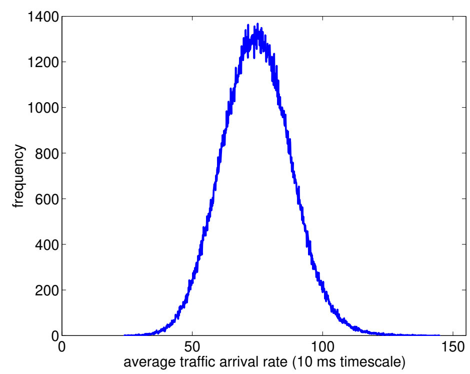
From (3.1) we see that the dominant characteristic which affects queuing delay is the marginal distribution of the traffic arrival process A at different time scales, t. In Chapter 2, we observed that backbone network traffic is aggregated from a large population of users, each of whom transmit data at a rate much smaller than the total link capacity. It is natural to expect, therefore, that the distribution of A_t is Gaussian as a result of the Central Limit Theorem. To determine if this is the case, we compute the distribution of A_t over a range of time scales t and apply a statistical test known as the Kolmogorov-Smirnov Test (K-S test) for normality [18] to test if the distribution is Gaussian.

The marginal distribution of the traffic arrival process at time scale t is computed by dividing a trace into non-overlapping blocks of duration t and computing the number of bits which arrive over each of these blocks (e.g. compute the number of bits which arrive over every 100 ms time interval). For purposes of presentation, we normalize A_t to get the average traffic arrival rate at each time scale t (i.e. we consider A_t/t).

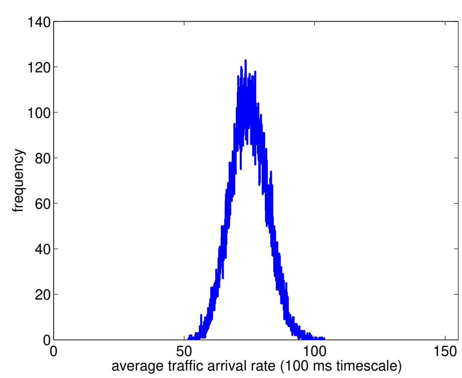
We first consider a one hour segment from 9:56 am to 10:56 am of trace WEST-04B. Figure 3.1 plots the marginal distribution WEST-04B at time scales of 1 ms, 10 ms, and 100 ms. At t=1 ms, the distribution appears Gaussian with mean 74.7 Mb/s and variance 852 $(Mb/s)^2$. At t=10 ms it appears Gaussian with mean 74.7 Mb/s and variance 178 $(Mb/s)^2$. At t=100 ms it appears Gaussian with mean 74.7 Mb/s and variance 49.5 $(Mb/s)^2$. Applying the K-S test to each of these distributions



(a) t = 1ms



(b) t = 10ms



(c) t = 100 ms

Figure 3.1: Marginal distribution of traffic arrivals for trace WEST-04B

confirms that they are consistent with a Gaussian distribution. When the test is applied to the distributions for larger time scales up to 60 seconds, it confirms that these distributions are also Gaussian.

Since a Gaussian distribution is fully specified by its mean and variance, to compute the delay probability (3.1) it is sufficient to know the mean and variance of A_t at each time scale t. The mean remains the same for all time scales as seen from Figure 3.1. The variance, however, changes from one time scale to the next. The relationship between the variance and time scale can be studied using a technique known as the variance-time (VT) plot. This is simply a plot of the variance versus the time scale t.

Before proceeding, it is important to note that prior studies have demonstrated that the VT plot may not provide much information about the structure of network traffic at time scales less than 100 ms. For example, [38] showed that for traffic measurements collected on an FDDI ring and various T3 links in the AT&T network, the behavior of the traffic arrival process over small time intervals could not be described only using second-order statistics (mean and variance). However, the measurements considered in [38] were of traffic with an average arrival rate between 1 Mb/s an 10 Mb/s, much lower levels of aggregation than the trace WEST-04B. From Figure 3.1 we can see that for large traffic aggregates, the distribution at small time scales is Gaussian and can therefore be fully described using only the mean and variance. The VT plot, therefore, provides enough information (in conjunction with the average traffic arrival rate) to completely characterize the traffic arrival process. Later in this section we will present an analysis of how much aggregation is needed before it is

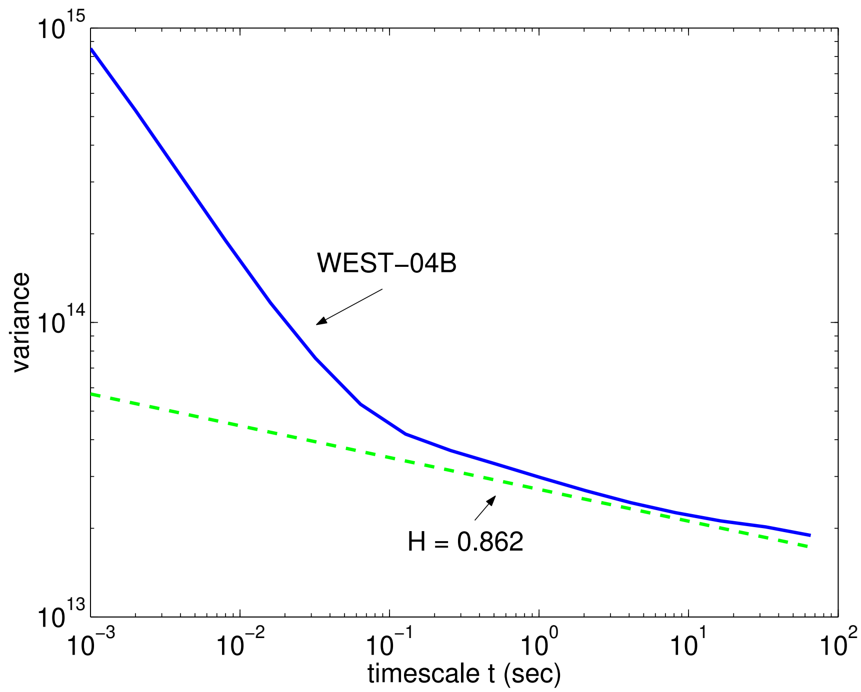


Figure 3.2: Variance-time plot for WEST-04B

possible to use second-order traffic models to describe the traffic arrival process. We will also address the statistical bias that may be introduced by using the VT plot to estimate the variance at small time scales.

The VT plot for the one hour segment of WEST-04B is shown in Figure 3.2. The figure shows that the variance exhibits a two-piece linear relationship with the time scale t. It decays quite rapidly over time scales between 1 ms and 75 ms, and starts to decay more slowly after that point. The slow decay of the variance at large time scales is indicative of a statistical property known as *long-range dependence* (LRD) which has been observed in LAN traffic [67] as well as WAN traffic [90], [38]. For traffic with LRD, the variance of the traffic arrival rate decays as a power of the time scale t

$$var(\frac{A_t}{t}) \sim t^{2H-2},\ as\, t \to \infty$$

where H is known as the Hurst parameter and takes a range of $0.5 < H < 1$. This behavior is quite different from Poisson traffic or other *short-range dependent* (SRD) traffic. SRD traffic has a Hurst parameter equal to 0.5, and the variance of A_t/t therefore decays exponentially with the timescale t. As a result SRD traffic "smooths out" as it is averaged over larger and larger time intervals, while LRD traffic remains quite "bursty".

To illustrate the difference between LRD and SRD traffic, we compare the one hour of traffic from WEST-04B with one hour of Poisson traffic with the same average traffic arrival rate as the WEST-04B traffic[1]. This comparison follows the same approach used in [67]. Figure 3.3 plots the traffic arrival rates at different time scales for the WEST-04B traffic and for the Poisson traffic. The top row of the figure plots the average traffic arrival rate over each one second interval during the one hour period. The second row plots the average traffic arrival rate over 100 ms intervals for the period between 450 seconds and 800 seconds indicated in black the top row of the figure. The bottom row plots the average traffic arrival rate over 10 ms intervals for the black region shown in the second row.

At the 10 ms timescale, the magnitude of fluctuations in WEST-04B are of the same magnitude as those for the Poisson traffic. However, when the Poisson traffic is averaged over larger and larger time scales, the variability of the traffic is greatly

[1] Poisson traffic is admittedly a very simplistic form of SRD traffic. More sophisticated SRD models, however, would show the same basic behavior as Poisson.

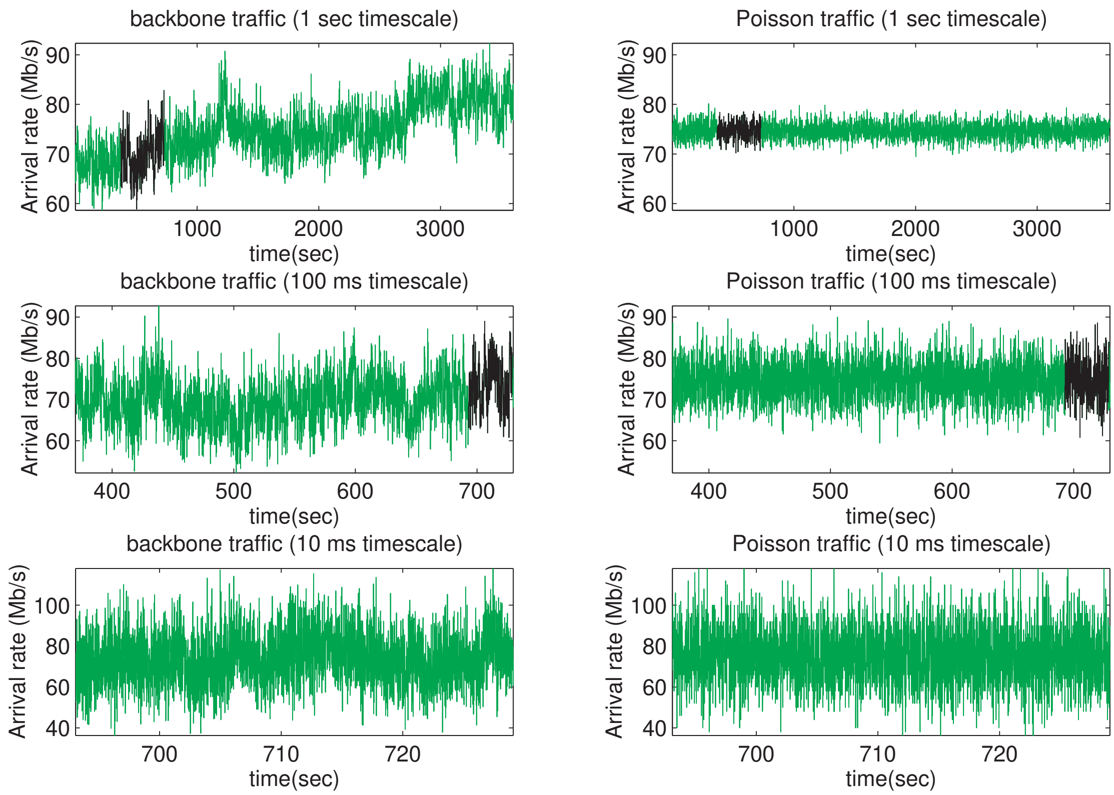


Figure 3.3: Comparison of WEST-04B traffic and Poisson traffic

reduced. The WEST-04B traffic, on the other hand, remains quite bursty even when averaged over one second intervals.

The LRD of network traffic has been shown to be the result of the distribution of individual user connection sizes [112]. [23] and [65] have shown that if the duration in time (or size in bytes) of individual user connections has a heavy-tailed distribution, the aggregate traffic will exhibit LRD. A heavy-tailed distribution is one in which $P(X > x) \sim x^{-\alpha}, 1 < \alpha < 2$, as $x \to \infty$. In fact, the Hurst parameter is directly related to the α parameter of the connection size distribution according to $H = (3 - \alpha)/2$ [112].

To validate that the user connection size distribution is responsible for the large time scale behavior of $var(A_t/t)$ observed in our measurements, we compute both H and α for the one hour segment of traffic from WEST-04B. One approach to estimating H is to compute the slope of the VT plot at large time scales [107]. However, this approach suffers from a statistical bias. For LRD traffic it is possible for the traffic arrival rate to exceed the mean arrival rate for a long period of time (i.e. to have a sustained burst of traffic over a long time interval). It is difficult to determine if the increase in traffic volume is the result of a shift in the mean arrival rate, or if it is simply a long burst of traffic. To address this problem, Abry and Veitch developed a wavelet based approach to estimating H which can reliably estimate H in the presence of level-shifts in the traffic [110]. We use this estimator, rather than the VT plot to estimate H. Using this estimator, at time scales greater than 100 ms, we find that $H = 0.862$ for WEST-04B. For reference, we plot the variance of a process with $H = 0.862$ as a dashed line in Figure 3.2. This closely matches the observed variance of the WEST-04B traffic.

We now compare this H value with the α parameter of the connection size distribution. We identify individual user flows in the WEST-04B measurement using the approach described in Section 2.3.4. There is one minor difference between the flow definition used in Chapter 2, and the flow definition used to generate Figure 3.4. In Chapter 2, the end of a flow was identified as the last packet after which no further packets between the same source and destination were observed for a 60 second interval. In Figure 3.4, we consider the end of a flow to be the last packet

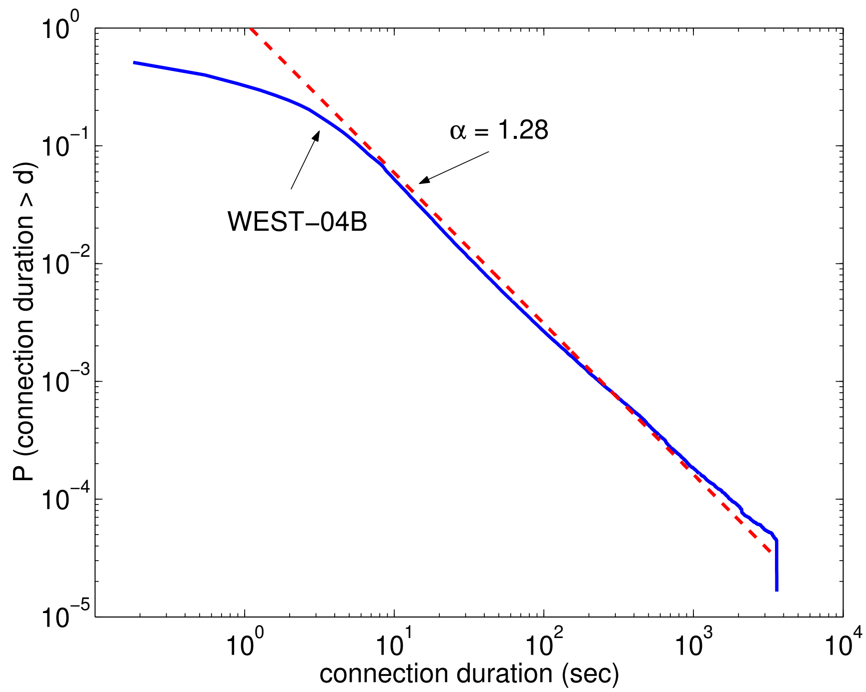


Figure 3.4: Connection size distribution for WEST-04B

after which there is an idle period of only 5 seconds. This definition is much closer to the idle period of 2 seconds used in [112]. Furthermore, if the idle period is increased to more than 5 seconds, the α parameter does not match the H parameter for this measurement.

Figure 3.4 plots the complementary distribution of the duration times of user connections. With $H = 0.862$, we expect this distribution to be heavy-tailed with $\alpha = 1.28$. The dotted line in Figure 3.4 shows a heavy-tailed distribution with parameter $\alpha = 1.28$. This closely matches the tail of the measured connection duration distribution. To further validate this relationship, we use the Hill estimator as described in [112] and the procedure described in [25] to estimate the actual α parameter for the connection size distribution. We find the actual connection size distribution is heavy-tailed with $\alpha = 1.30$ which indicates H should be 0.85. This value of H is within the confidence intervals of the wavelet-based estimator.

Next we investigate the behavior of the variance at time scales less than 75 ms. We see from Figure 3.2 that the variance at small time scales also has a linear relationship with t, but the slope is much larger and the variance is much higher than can be explained by the connection size distribution. The reason for this is that the theory relating the connection size distribution to H considers user connections to be constant bit rate (CBR). In a real network, however, user connections are far from CBR. In Chapter 2 we observed that over 90% of the traffic in the network is transmitted using TCP. TCP connections transmit a burst of packets corresponding to the TCP window size, wait one round-trip-time (rtt) for the acknowledgment, and then transmit another burst of packets. At time scales greater than the rtt, the connections can be approximated as CBR streams with a rate of one window per rtt. While there can be a variation of the window size over time, it has been empirically demonstrated that the CBR approximation is reasonable [112]. However, as the time scale falls below the rtt, individual TCP connections become much more variable than CBR streams resulting in the higher variance.

A direct relationship between the rtt and the break point between the two scaling regions of the VT plot has been demonstrated though the use of simulation [37]. This study performed a simulation where all connections had a rtt of 24 ms and a second simulation where all connections had a rtt of 610 ms. They found that the linear relationship between the variance and timescale which was observed at large time

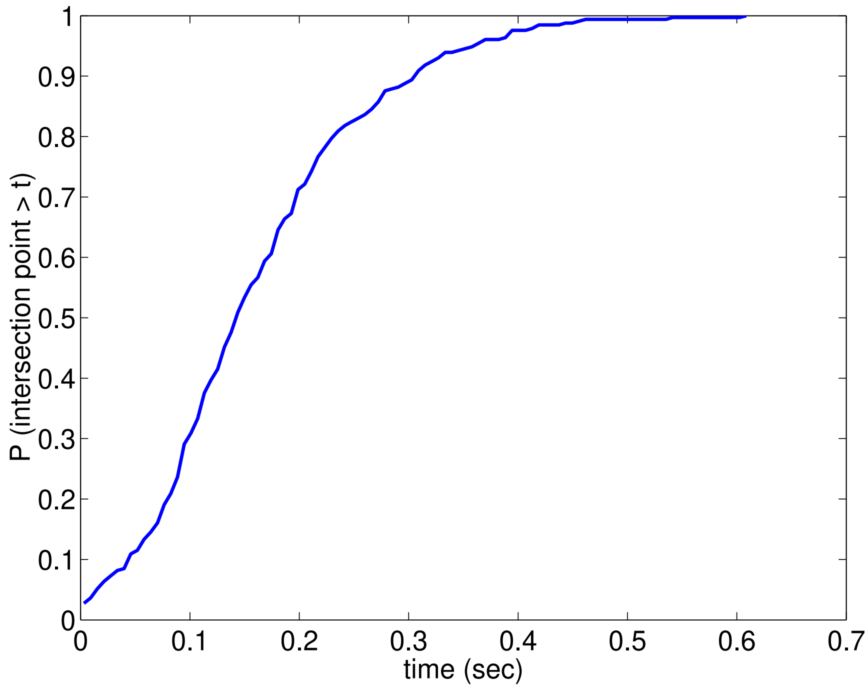


Figure 3.5: Distribution of VT plot transition point

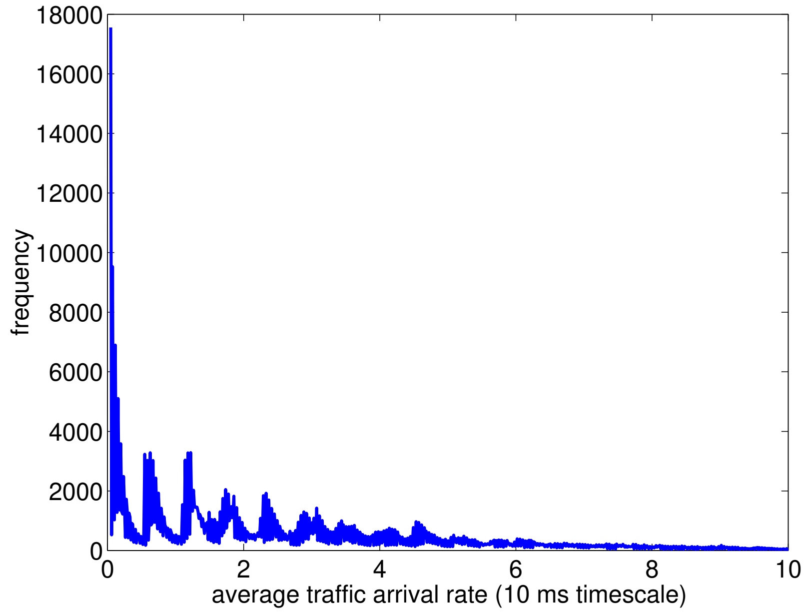
scales (i.e. the relationship due to the connection size distribution) broke down at a time scale just above the rtt of the user connections. In general, for each of the 331 one-hour measurements we study, we find that the transition point occurs "near" the median rtt of the connections observed in the traces. Figure 3.5 plots the cumulative distribution of the time scale at which the transition point occurs for all 331 traces. This distribution closely matches the distribution of median rtts which were shown in Figure 2.16. For the WEST-04B trace in particular, the median rtt is 96.9 ms which is approximately the point at which the variance begins to rapidly increase.

However, we do not find a statistically significant correlation between the median or mean rtt and the breakpoint location. In general the rtt distribution is quite complex, and the mean or median value is insufficient to fully describe the distribution.

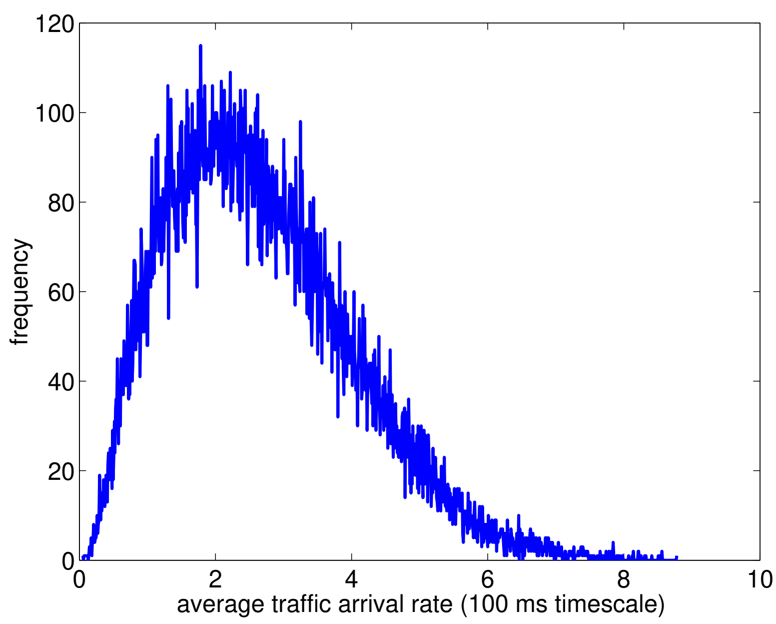
As a result, we are unable to fully explain the exact location of the breakpoint and the cause of the linear behavior in the variance at small time scales. However, we are able to develop a model to capture this behavior and compute the resulting queuing delays.

Before developing the model, it is interesting to compare the small time scale behavior observed in our measurements with the measurements used in prior studies. As mentioned earlier, prior measurements exhibited quite complex distributions at small time scales and could therefore not be described using only second order properties. To illustrate this, we consider one of the measurements used in these prior studies. This measurement is known as *DEC-WRL-2* and was collected on a 100 Mb/s Ethernet segment connected to the primary connection between DEC's Western Research Lab and the Internet [90]. Figure 3.6 plots the marginal distribution of A_t/t at 10 ms intervals and 100 ms intervals for *DEC-WRL-2*[2] . Comparing these distributions to the distributions for WEST-04B shown in Figure 3.1, we see that, indeed, the distribution of the traffic arrivals at small time scales for DEC-WRL-2 is much more complex than the Gaussian distributions seen for WEST-04B. In particular, the distribution at the 10 ms time scale has noticeable spikes at 50 kb/s, 550 kb/s, 1150 kb/s, and so on. The reason for these spikes is that the average traffic arrival rate for the *DEC-WRL-2* trace is only 250 Kb/s. Assuming an average packet size of 300 bytes, this corresponds to an average rate of approximately one packet every 10 ms. The complex distribution is the result of observing some intervals with

[2]We are unable to investigate the distribution at 1 ms timescales since the measurement data for DEC-WRL-2 does not have enough timescale granularity.



(a) t = 10ms



(b) t = 100 ms

Figure 3.6: Marginal distribution at $t = 10$ms for *DEC-WRL-2*

a single packet, some intervals with two packets, and so on.

These complex distributions at small time scales have been observed in many other traffic studies [67], [92], [38], [34], [97]. For traffic with such complex distributions, it is inappropriate to use only second-order statistics to describe the distribution. However, these studies consider traffic whose average arrival rate ranges from several hundred kb/s to at most 10 Mb/s. This represents traffic from a relatively small number of users (e.g., 10,000 user connections over a one hour period for the measurement used in [112]). While at the time these measurements were collected, they represented a large traffic volumes, traffic in today's network is orders of magnitude larger. In our traces, the average arrival rate ranges from 1 Mb/s to over 1.5 Gb/s. For trace WEST-04B in particular, the average traffic rate is almost 75 Mb/s, and there are nearly 5 million unique user connections over the one hour period. While the *DEC-WRL-2* traffic would transmit an average of one packet per 10 ms time interval, the WEST-04B traffic has an average of 235 packets in each 10 ms interval. As a result of the increased aggregation the marginal distributions are approximately Gaussian at small time scales and can be completely described by second order statistics.

A natural question to ask is: how much aggregation is needed before the marginal distribution at small time scales become Gaussian? We can investigate this by considering the complete set of 331 one hour traffic measurements. For some of the measurements (especially for those collected between 1:00 am and 4:00 am), the average traffic arrival rate is quite low. In some cases it is as low as 1 Mb/s. For other measurements collected during afternoon hours on highly utilized links, the traffic

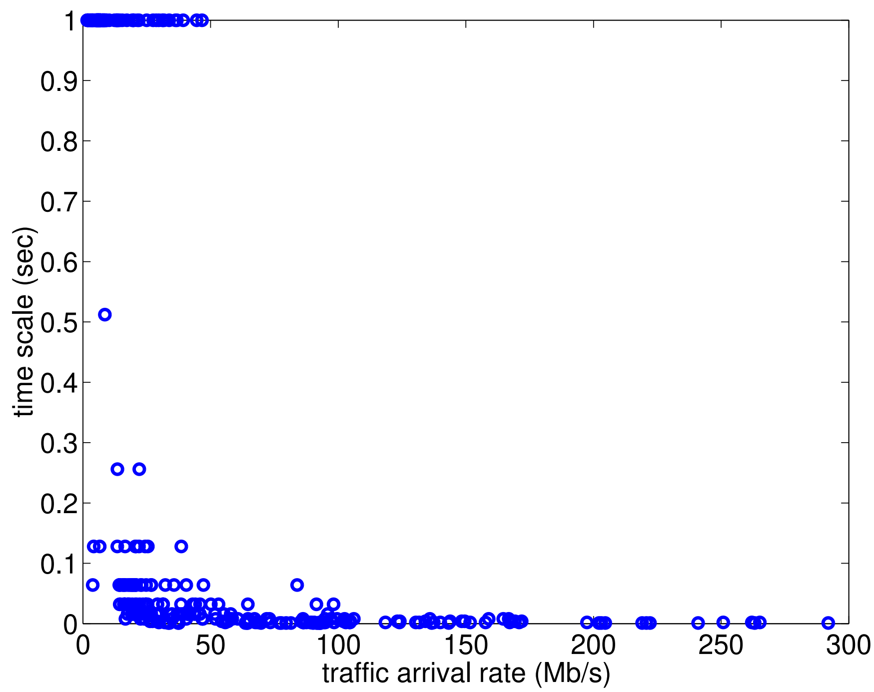


Figure 3.7: Minimum time scale at which marginal distributions are Gaussian

volume can reach almost 300 Mb/s. Using the K-S test, we can determine which of these traces have Gaussian marginal distributions at small time scales, and which have the more complex distributions.

More precisely, we compute the marginal distribution of the traffic arrival process at time scales from 1 ms to 1 sec for each of the traces. At each time scale we apply the K-S test to determine if the distribution is Gaussian. For each trace we find the smallest time scale at which the marginal distribution is Gaussian.

Figure 3.7 plots the minimum Gaussian time scale against the mean arrival rate of the traffic for each of the traces. We see that for all but four traces with traffic volume greater than 50 Mb/s, the minimum time scale is between 1 ms and 8 ms. These traces have characteristics very similar to those shown for WEST-04B. Below 50 Mb/s there is much more variation. Two-thirds of the traces with traffic volume between 5 Mb/s and 50 Mb/s have a minimum Gaussian time scale between 1 ms and 64 ms, while one-third exhibit distributions similar to those shown for the *DEC-WRL-2* measurement. For traces with traffic volume less than 5 Mb/s, the marginal distributions are never Gaussian. All of these low volume traffic measurements resemble the *DEC-WRL-2* traffic. Therefore, if traffic volume on a link is less than 5 Mb/s, and for some traffic between 5 Mb/s and 50 Mb/s, it is inappropriate to use only second order statistics to describe the traffic arrival process at small time scales. During the busy hour of the day, however, traffic volume on nearly all backbone links is greater than 50 Mb/s and Gaussian models are therefore appropriate to use.

There are situations where 50 Mb/s traffic will not have enough aggregation to use Gaussian models. Consider, for example, a link carrying three 20 Mb/s HDTV video streams. The bandwidth guidelines we present are only valid for the traffic with the same mix of user connections that we see in today's backbone. In particular, the traffic must be aggregated from a large population of users, and the rate of an individual user should be much less than the rate of the total traffic aggregate.

3.3 Modeling Network Traffic

Modeling network traffic which is aggregated from a small number of users has proven to be quite difficult. In the previous section, we saw that the low volume traffic measurements used in prior studies have very complex distributions for the traffic arrival process at time scales less than 100 ms. Two basic approaches have been taken to model these complex distributions. One approach is to simply assume that the distributions are Gaussian. For the prior traffic measurements, at time scales greater than 100 ms, the traffic arrival process *is* approximately Gaussian, and the variance of the Gaussian distribution exhibits a linear relationship with the time scale as shown by the dashed line in Figure 3.2. One could therefore assume that the distributions at small time scales are also Gaussian and that the linear relationship also holds. Such a model is known as Fractional Brownian Motion (FBM), and it was originally proposed as a model for network traffic by Norros [83].

However, in earlier measurements, the distribution of A_t/t at small time scales is clearly not Gaussian as seen from Figure 3.6. Furthermore, studies have shown that the small time scale distributions can be the dominant factor of queuing delay [100],[53],[47]. Therefore, many researchers have developed models which accurately represent the complex distributions of the traffic arrivals at small time scales. The most common approach is to use a construction known as a multifractal cascade. Such models have been proposed in [38],[98],[97],[34]. However, because these models are trying to represent a very complex distribution, they require a large number of parameters. As a result, the models are difficult to use analytically, and they require collection of detailed packet-level measurements in order to estimate the model parameters. With the packet-level measurements, one could just perform the simulations described in Chapter 2 to evaluate the queuing delay.

As seen in the previous section, backbone network traffic which is aggregated from a large population of users does not exhibit the same complex traffic arrival distributions at time scales between 1 ms and 100 ms[3]. As a result, we can use much more straightforward traffic models to represent aggregate traffic.

3.3.1 Two-Scale Fractional Brownian Motion

To model backbone traffic we would like a process which has Gaussian marginals with a variance that obeys the two-piece linear relationship observed in the traces. This can be accomplished using an extension of traditional Fractional Brownian Motion known as Multiscale FBM. $(M_K) - FBM$, is an extension of FBM with a Hurst parameter that varies at different time scales. We can therefore use one Hurst parameter, H_0, for large time scales and another Hurst parameter, H_1, for small time scales. $(M_K) - FBM$ was originally used by Benassi and Deguy [5] for image synthesis and Bardet and Bertrand [4] to model biomechanic data.

It is important to note that there are several different processes which have Gaussian marginals and a variance that obeys a two-piece linear relationship with the time scale. For example, one could consider traditional FBM with a periodic component. The purpose of this study, however, is to evaluate queuing delay. We are therefore interested in finding a model which has the same marginal distributions as that observed in the network traffic. Multiscale-FBM is one such model. All other models with the same behavior of the marginal distributions will give identical results when evaluating the queuing delay.

[3]It should be noted that at time scales of several hundred μs, aggregate traffic has complex distributions similar to those shown at the 10 ms time scale for *DEC-WRL-2*. However, since we are interested in queuing delays on the order of several milliseconds we do not need to consider the behavior of the traffic at microsecond time scales.

To construct Multiscale FBM, we start with the harmonizable representation of the traditional FBM process, $B_H(t)$ [101]

$$B_H(t) = \int_{-\infty}^{\infty} \frac{e^{i\omega t} - 1}{C(H)\left|\omega\right|^{H+1/2}} \widetilde{W}(d\omega)$$

$W(dx)$ is a Brownian measure and $\widetilde{W}(d\omega)$ is its Fourier transform, and $C(H) = \frac{\pi}{H\Gamma(2H)sinH\pi}^{1/2}$. For such a process H is constant for all frequencies (inverse of time scale). $(M_K) - FBM$ is a generalization of this process in which H is varies across different frequencies. We define an $(M_K) - FBM$, $X_\eta(t)$, as a process such that

$$X_\eta(t) = \int_{-\infty}^{\infty} \frac{e^{i\omega t} - 1}{\eta(\omega)} \widetilde{W}(d\omega)\ , -\infty < t < \infty$$

where

- $K \in N$, represents the number of Hurst parameters
- for $i = 0, 1, \ldots, K$ there exist $(\omega_i\ , a_i\ , H_i\ ) {\in} (R_+,\ R_+,\ (0.5, 1))$ such that $\eta(\omega) = \frac{C(H_i)|\omega|^{H_i+1/2}}{\sqrt{a_i}}$ for $\omega_i \leq \omega < \omega_{i+1}$ with $0 = \omega_0 < \omega_1 < \cdots < \omega_K < \omega_{K+1} = \infty$
- $\eta(-\omega) = \eta(\omega)$

[4] has shown that $X_\eta(t)$ is a Gaussian process with stationary increments and variance at time scale δ, $var(\delta) = E[X_\eta(t + \delta) - X_\eta(t)]^2$, given by

$$var(\delta) = 4 \sum_{i=0}^{K} \delta^{2H_i} \frac{a_i}{C(H_i)^2} \int_{\delta\omega_i}^{\delta\omega_{i+1}} \frac{1 - cos\upsilon}{\upsilon^{2H_i+1}} d\upsilon. \tag{3.2}$$

To derive the queue length distribution for this process, we follow the same procedure Norros used to derive the queue length distribution for FBM [83]. Let A_η be the cumulative traffic arrival process

$$A_\eta(t) = mt + \sqrt{m}X_\eta(t),$$

where m is the mean arrival rate of the traffic, and the term $\sqrt{m}X_\eta(t)$ describes the fluctuations around the mean.

We use the lower bound (3.1) to compute the queue length distribution

$$P[Q > x] \geq \sup_{t \geq 0} P[A_\eta(t) > x + Ct] \tag{3.3}$$

Since at time scale t, $A_\eta(t)$ has a Gaussian distribution with mean mt and variance $m \cdot var(t)$ the queue length distribution is

$$P[Q > x] = \sup_{t \geq 0} \overline{\Phi}(\frac{x + Ct - mt}{\sqrt{m \cdot var(t)}}) \tag{3.4}$$

Where $\overline{\Phi}$ is the residual distribution function of the standard Gaussian distribution. In the general case for $M_k - FBM$ with many Hurst parameters, finding the t which maximizes the right hand side of (3.4) is difficult. However, as we have seen from our measurements, our traffic has only two distinct scaling regions. We therefore consider the specific the case when $K = 1$ and call this *two-scale FBM*. In this case we can use the following approximation

$$var(\delta) = \begin{cases} \delta^{2H_1} \frac{a_1}{C(H_1)^2} & , 0 \leq \delta < \frac{1}{\omega_1} \\ \delta^{2H_0} \frac{a_0}{C(H_0)^2} & , \frac{1}{\omega_1} < \delta < \infty \end{cases} \tag{3.5}$$

where (a_1, H_1) represent the linear region at small time scales, (a_0, H_0) represent the linear region at large time scales, and $\frac{1}{\omega_1}$ is the transition point between the two regions.

Using this form for the variance, we find that (3.4) is maximized at

$$t = t^* = \begin{cases} \frac{H_1}{1-H_1} \frac{x}{C-m} & , x < x_c \\ \frac{H_0}{1-H_0} \frac{x}{C-m} & , x \geq x_c \end{cases}$$

and the queue length distribution is

$$P[Q > x] \sim \begin{cases} \exp(-\kappa(a_1, H_1)) x^{2-2H_1} & , x < x_c \\ \exp(-\kappa(a_0, H_0)) x^{2-2H_0} & , x \geq x_c \end{cases} \tag{3.6}$$

where

$$\kappa(a, H) = \frac{(C-m)^{2H}}{2am(1-H)^{2-2H}(H)^{2H}}$$

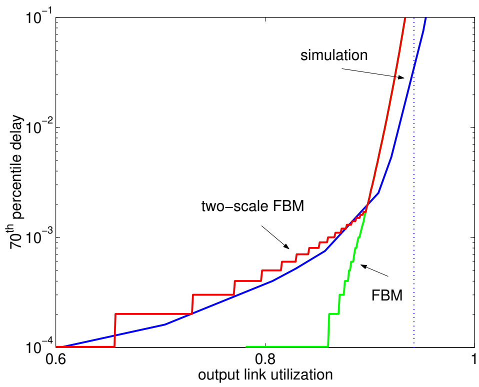
$$x_c = \frac{(C-m)(1-H_1)}{H_1} e^{\frac{H_0 \log(\frac{H_1}{H_0}) + (H_0 - 1)\log(\frac{H_1 - 1}{H_0 - 1}) + \frac{1}{2}\log(\frac{a_1}{a_0})}{H_0 - H_1}}$$

3.3.2 Model Validation

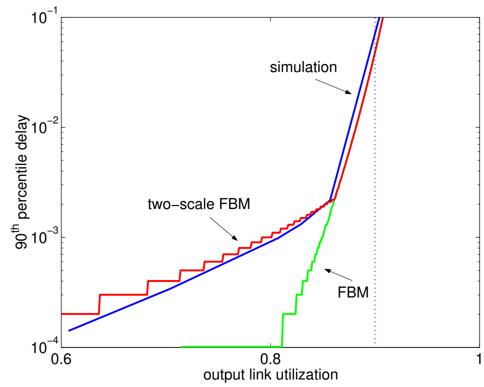
To validate the two-scale FBM model we compare the actual delay experienced by the measured traffic with the delay computed using the model. To determine the actual delay for the traffic we use a queuing simulator as described in Chapter 2.4.6.

To compute the delay using two-scale FBM, we must estimate the model parameters from our traces. We use the Abry-Veitch estimator [110] to determine the H_0 and H_1 parameters and we use linear regression on the variance-time plot to estimate a_0 and a_1. However, we do not know, a priori, over which time scales to estimate (a_0, H_0) and over which time scales to estimate (a_1, H_1). As we have seen in the previous section, the breakpoint between the two regions of the model typically occurs at time scales between 100 ms and 500 ms. We therefore do not consider this region and estimate (a_1, H_1) from the traffic characteristics at time scales of 2 ms - 64 ms, and we estimate (a_0, H_0) from the traffic characteristics at time scales of 512 ms - 2 min.

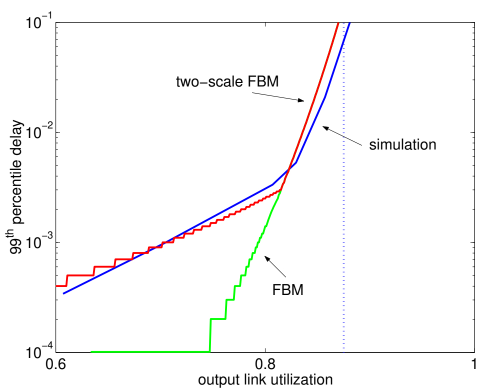
We first investigate how well the model estimates the delay for one hour of the WEST-04B trace. The parameters for this trace are $H_1 = 0.62$, $H_0 = 0.89$, $a_1 = 69.6$ kb·s, $a_0 = 338$ kb·s, and $m = 75$ Mb/s. We compare the delay distribution obtained using the model and the delay distribution obtained from the simulation for a range of output link utilizations, ρ. The results are shown in Figure 3.8. The vertical dotted line shows the point at which the maximum delay in the simulation exceeds 250 ms. Beyond this point it is possible to experience loss in the actual network the simulation does not capture the effects introduced by the TCP congestion control mechanism.



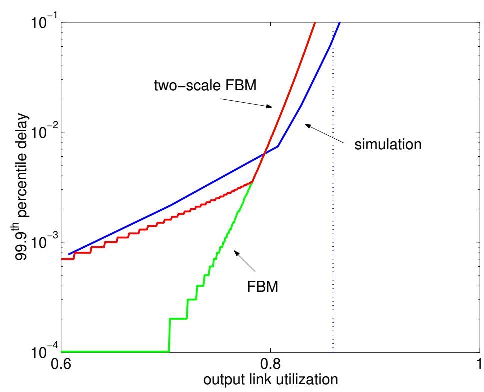
(a) 70th percentile of the delay distribution



(b) 90th percentile of the delay distribution



(c) 99th percentile of the delay distribution



(d) 99.9th percentile of the delay distribution

Figure 3.8: Simulation delay and two-scale FBM delay for WEST-04B

For a reference, we also show the delays that are predicted using the standard FBM model.

Figure 3.8 plots the 70th, 90th, 99th and 99.9th percentile of the delay distribution at different output link utilizations. Similar results continue at percentiles up to 99.99%. Percentiles below 70% do not perform as well. The reason for this is the approximation used in (3.1) is only valid for the tail of the delay distribution. We find it works well at percentiles above 70%. Fortunately, this is the area of interest when designing a network to meet the requirements of latency sensitive traffic. Applications such as voice or video can only tolerate a small percentage of their packets which exceed the delay requirement.

From the figure we see that the traditional FBM and the 2-scale FBM models perform the same when the output utilization is high. In this region, the large time scale characteristics dominate the queuing performance (i.e., t^* is greater than several hundred milliseconds). Both FBM and the 2-scale FBM are accurate models for the large time scale characteristics, so they perform the same. At low utilization, the 2-scale FBM model performs much better than traditional FBM. In this region t^* is less than several hundred milliseconds. Since 2-scale FBM is a much better model for the small time scale characteristics, the delay estimate is much more accurate.

Next we evaluate the model performance for the rest of the traces. It is not possible to repeat Figure 3.8 for all traces. Instead we evaluate the model performance at a link utilization $\rho = 0.7$ and at $\rho = 0.9$. The performance at $\rho = 0.7$ determines how well the model fits before the knee of the curve shown in Figure 3.8, and the performance

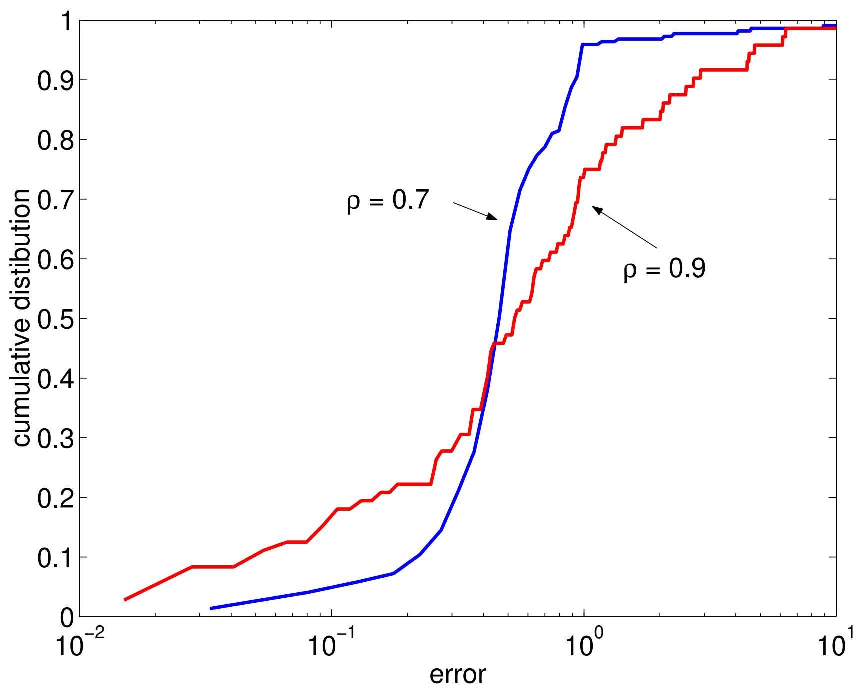


Figure 3.9: Two-scale FBM performance for 300 sample traces, $\epsilon = 0.001$

at $\rho = 0.9$ is indicative of how well the model fits after the knee. $\rho = 0.9$ may not be a reasonable operating point for a commercial network as the delays are quite large, but we would like to evaluate the model performance in this region. We only show results for $\epsilon = 0.001$ as this value showed the worst performance for WES-04B and most of the other traces.

Figure 3.9 plots the difference in the delay estimated by the 2-scale FBM model and the delay obtained in simulation: $error = \frac{|d_{2-scaleFBM} - d_{simulator}|}{d_{simulator}}$. From the Figure we see that at $\rho = 0.7$, 80% of the flows have an error less than 0.75, and 96% of the flows have an error of less than 1. An error of 1 may seem to be quite large (100% error). However, in terms of actual delay, it represents a difference between 1 ms and 2 ms or 4 ms and 8 ms. For a reference, at $\epsilon = 0.001$ and $\rho = 0.7$, the WEST-04B trace shown in Figure 3.8 has $d_{simulator}$ = 2.2 ms and $d_{2-scaleFBM}$ = 1.6 ms. This corresponds to an error of 0.37, close to the median error for all traces.

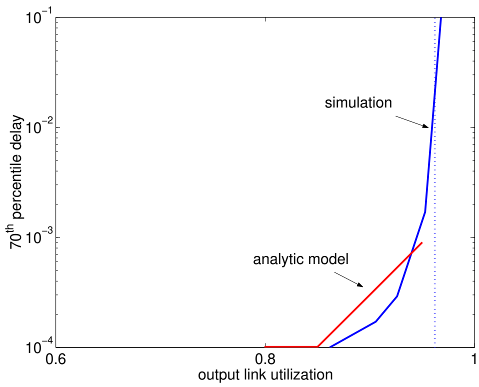
At $\rho = 0.9$ the model does not appear to perform as well. Only 75% of the flows have an error of less than 1. However, consider Figure 3.8(d). Due to the rapid increase in delay, shifting one of the curves to the left or right can result in a very large difference between the two delay values. In fact, for WEST-04B, the error at $\rho = 0.9$ and $\epsilon = 0.001$ is almost 10, one of the highest errors of all traces considered. We can consider the results shown in Figure 3.8(d) to be among the worst of all traces we have studied. Furthermore, from a bandwidth provisioning point of view, the location of the knee of the curve shown in Figure 3.8 is the most important aspect rather than the actual magnitude of the delay above the knee. The model does accurately predict this knee location.

3.3.3 Queue Fed by Multiple Two-scale FBM Flows

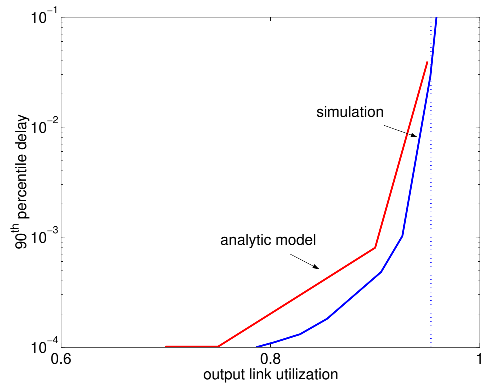
To derive the queue length distribution of a queue fed by multiple 2-scale FBM flows, we follow the same procedure as in Section 3.3.1. Consider a queue fed by N two-scale FBM flows. Let A_t^n be a 2-scale FBM process corresponding to flow n. The queue length distribution is

$$P[Q > x] = \sup_{t \geq 0} P(\sum_n A_t^n > x + Ct)$$

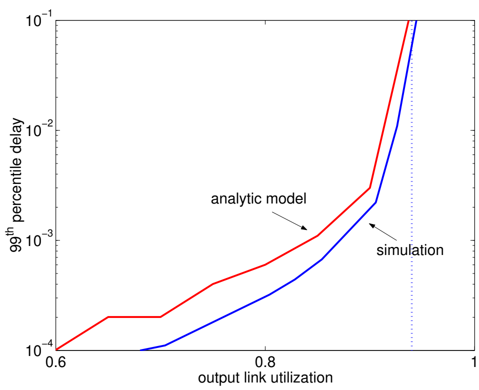
Assuming the flows are independent, at each time scale, t, the distribution of $\sum_n A_t^n$ is Gaussian with mean $\sum_n m_n t$ and variance $\sum_n m_n \cdot var_n(t)$. The queue length



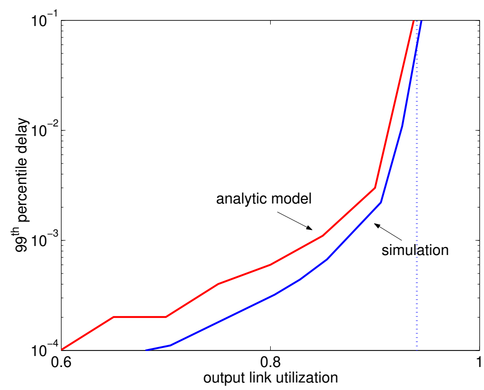
(a) 70th percentile delay of the delay distribution



(b) 90th percentile of the delay distribution



(c) 99th percentile of the delay distribution



(d) 99.9th percentile of the delay distribution

Figure 3.10: Delay for a queue with two flow inputs

distribution is then

$$P[Q > x] = \sup_{t \geq 0} \overline{\Phi}\left(\frac{x + Ct - \sum_n m_n t}{\sqrt{\sum_n m_n \cdot var_n(t)}}\right) \tag{3.7}$$

Unlike the single flow case, we cannot use the variance approximation (3.5) because the variance of the aggregated flow has more than two scaling regions. As a result, (3.7) cannot be further simplified. While analytically cumbersome, (3.7) can be easily computed using Matlab or C programs.

To validate (3.7), we perform simulations of a queue fed by two flows. We consider the first flow to correspond to the one hour segment of WEST-04B and the second flow to correspond to a one hour segment of WEST-05B. Both of these traces were collected on input links to the same router over the same time interval. Our measurement systems are synchronized to within 5 μs using GPS, so it is reasonable to consider both of these measurements as representing the input to the same queue. The delay obtained through simulation and the delay computed analytically are shown in Figure 3.10. The results for a queue fed by two two-scale FBM flows are very similar to that for a queue fed by a single two-scale FBM flow shown in Figure 3.8. Repeating this experiment with more than two flows produced similar results.

3.4 End-to-end Delay

In the previous section we developed a method to compute the delay distribution for a single queue. Now we address the question of how to compute the end-to-end queuing

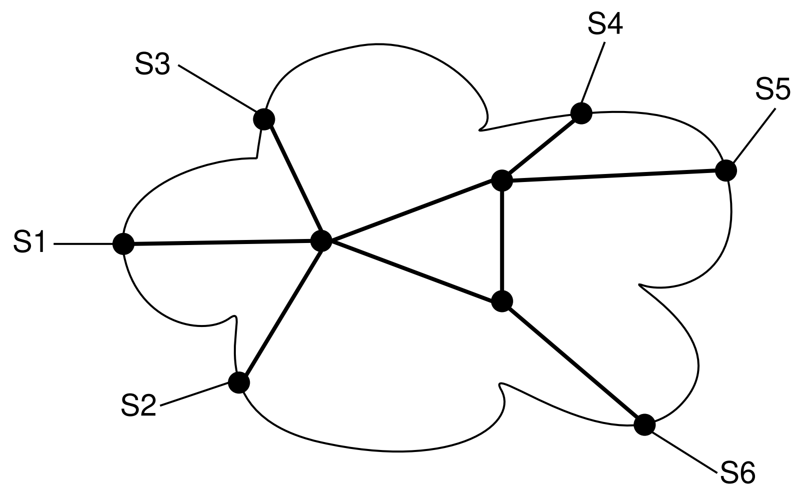


Figure 3.11: Network topology used for simulation

delay through a network. Consider the sample network shown in Figure 3.11. For this network, we would like to compute the end-to-end delay between any ingress and egress point. To do this, we use the following procedure. First, we model the traffic flow between each ingress and egress point using a two-scale FBM process. Then, for each link in the network we determine which ingress/egress flows arrive at the link and compute the delay experienced in that queue. The end-to-end delay over a path in the network is found by convolving the delay distributions for every queue along the end-to-end path.

In order to use this procedure we must make two assumptions. First, we assume that the characteristics of a flow remain the same throughout the network. Second, we assume that the delays at each queue are independent and the end-to-end delay can therefore be obtained by convolution. In this section we validate that these two assumptions are reasonable, and we compare the end-to-end delays obtained through this analytic method with the end-to-end delays seen in simulation.

3.4.1 Network Decoupling

The end-to-end delay computation is greatly simplified if the same parameters for a flow can be used at each hop the flow traverses. While it is not possible to do this in all networks, in many practical situations it is reasonable to consider the characteristics of a flow are unchanged throughout the network. The most well known example of this behavior is *Kleinrock's independence approximation* [64]. Under many reasonable network scenarios, if a queue is fed by multiple Poisson input streams, the output of the queue is also Poisson. Similar results have been derived for traffic such as FBM which exhibits a so-called *Large Deviations Principle* [113]. These arguments have also been used to justify that the effective bandwidth of a flow remains unchanged throughout the network [30]. The basic idea behind these arguments is that in many network scenarios, queues have sufficient output capacity so that very little queuing occurs for most packets. Only a small fraction of the traffic will experience large queuing delays. As a result, the flows passing through the queue will not be affected by the queuing. Such network conditions will be found in the network scenarios in which we are interested. We are interested in networks which are designed to meet a delay requirement which specifies only a small fraction of the packets experience long queuing delays.

To verify that this is a reasonable assumption for the network scenarios we consider we perform simulations using the network shown in Figure 3.11. We use a set of one hour measurements from six traces: WEST-16A, EAST2-13A, EAST1-08B, EAST2-11B, EAST1-09A, and EAST1-10B. All of these measurements were collected over the

Node	Destination Addresses
S1	0.0.0.0-127.255.255.255
S2	128.0.0.0-191.255.255.255
S3	192.0.0.0-199.255.255.255
S4	200.0.0.0-207.255.255.255
S5	208.0.0.0-215.255.255.255
S6	216.0.0.0-223.255.255.255

note: addresses 224.0.0.0 - 255.255.255.255 are the multicast and reserved address range. We do not observe any packets with these destination addresses.

Table 3.1: Mapping between destination IP address and network egress link

same one-hour interval. We use these traces as the inputs to the network, $S1$ - $S6$, respectively. To generate traffic demand between each ingress and egress point, we subdivide each trace into six separate flows according to the destination IP address of the packets in the trace. The mapping between IP address and egress node is shown in Table 3.1. The two-scale FBM parameters for each of the ingress/egress flows is given in Table 3.12.

We find that all but three of the sub-traces have sufficient aggregation to be modeled using 2-scale FBM. However, these three traces have an average rate of 0.81, 1.51, and 1.94 Mb/s. Since these flows are so small, we find there is little difference between the end-to-end delay computed when we model these flows as 2-scale FBM and when we completely ignore these flows in the computation. We therefore do not consider them in the computation. We do, however, include these flows in the simulation results.

To test if the flow parameters are unchanged in the network, we simulate the network and compute the model parameters for each flow when it enters the network

	S1	S2	S3	S4	S5	S6
S1	m = 136 H_1 = 0.577 a_1 =78.8 H_0 =0.851 a_0 =269	m = 37.2 H_1 = 0.594 a_1 = 110 H_0 = 0.884 a_0 = 370	m = 14.7 H_1 = 0.626 a_1 = 132 H_0 = 0.803 a_0 = 435	m = 16.0 H_1 = 0.562 a_1 = 59.6 H_0 = 0.895 a_0 = 266	m = 7.95 H_1 = 0.599 a_1 = 75.3 H_0 = 0.885 a_0 = 280	m = 6.90 H_1 = 0.582 a_1 = 90.8 H_0 = 0.936 a_0 = 423
S2	m = 82.6 H_1 = 0.701 a_1 = 107 H_0 = 0.880 a_0 = 141	m = 1.51 H_1 = 0.728 a_1 = 608 H_0 = 0.567 a_0 = 659	m = 3.51 H_1 = 0.632 a_1 = 189 H_0 = 0.868 a_0 = 389	m = 6.78 H_1 = 0.566 a_1 = 12.2 H_0 = 0.905 a_0 = 27.11	m = 6.21 H_1 = 0.568 a_1 = 36.6 H_0 = 0.705 a_0 = 95.8	m = 1.94 H_1 = 0.706 a_1 = 123 H_0 = 0.883 a_0 = 249
S3	m = 38.8 H_1 = 0.588 a_1 = 68.5 H_0 = 0.858 a_0 = 254	m = 25.6 H_1 = 0.584 a_1 = 48.8 H_0 = 0.897 a_0 = 151	m = 12.8 H_1 = 0.593 a_1 = 53.0 H_0 = 0.948 a_0 = 149	m = 18.8 H_1 = 0.635 a_1 = 106 H_0 = 0.974 a_0 = 373	m = 20.0 H_1 = 0.590 a_1 = 40.3 H_0 = 0.871 a_0 = 78.7	m = 15.7 H_1 = 0.579 a_1 = 31.4 H_0 = 0.818 a_0 = 56.3
S4	m = 49.9 H_1 = 0.500 a_1 = 65.3 H_0 = 0.999 a_0 = 477	m = 42.5 H_1 = 0.500 a_1 = 71.1 H_0 = 0.978 a_0 = 505	m = 16.9 H_1 = 0.564 a_1 = 117 H_0 = 0.953 a_0 = 530	m = 28.6 H_1 = 0.520 a_1 = 43.7 H_0 = 0.864 a_0 = 217	m = 18.5 H_1 = 0.519 a_1 = 30.1 H_0 = 0.894 a_0 = 116	m = 14.3 H_1 = 0.556 a_1 = 101 H_0 = 0.969 a_0 = 497
S5	m = 4.72 H_1 = 0.620 a_1 = 27.3 H_0 = 0.992 a_0 = 67.0	m = 5.29 H_1 = 0.614 a_1 = 29.8 H_0 = 0.926 a_0 = 103	m = 7.30 H_1 = 0.621 a_1 = 54.3 H_0 = 0.944 a_0 = 121	m = 0.871 H_1 = 0.506 a_1 = 4.74 H_0 = 0.879 a_0 = 24.4	m = 143 H_1 = 0.553 a_1 = 31.6 H_0 = 0.896 a_0 = 95.9	m = 47.2 H_1 = 0.549 a_1 = 33.9 H_0 = 0.913 a_0 = 92.8
S6	m = 56.9 H_1 = 0.503 a_1 = 18.4 H_0 = 0.852 a_0 = 96.2	m = 29.7 H_1 = 0.535 a_1 = 24.9 H_0 = 0.833 a_0 = 116	m = 14.3 H_1 = 0.516 a_1 = 19.0 H_0 = 0.824 a_0 = 80.0	m = 22.3 H_1 = 0.521 a_1 = 21.2 H_0 = 0.835 a_0 = 113	m = 37.7 H_1 = 0.530 a_1 = 27.5 H_0 = 0.776 a_0 = 130	m = 18.1 H_1 = 0.507 a_1 = 27.1 H_0 = 0.921 a_0 = 185

units: m (Mb/s), a_1, a_0 (kb·s)

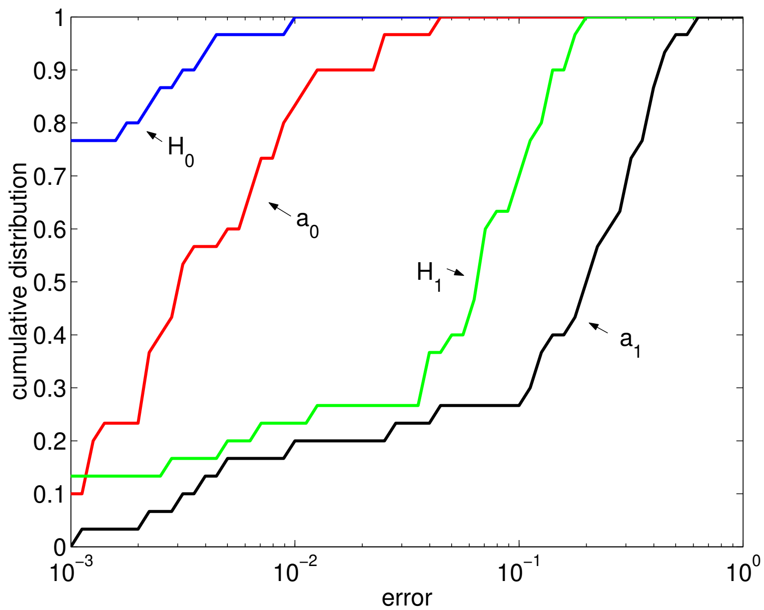
Table 3.2: Traffic demand matrix for simulation

at the ingress node and when it departs the network at the egress node. For each flow, we compute the percent difference between the parameters at the ingress and the parameters at the egress. Figure 3.12 plots the cumulative distribution for all flows. We performed a simulation where all links had a utilization of ρ=0.7 and a second simulation with ρ =0.95[4]. Since the mean arrival rate of the flows are fixed, to change the link utilization we change the link capacity.

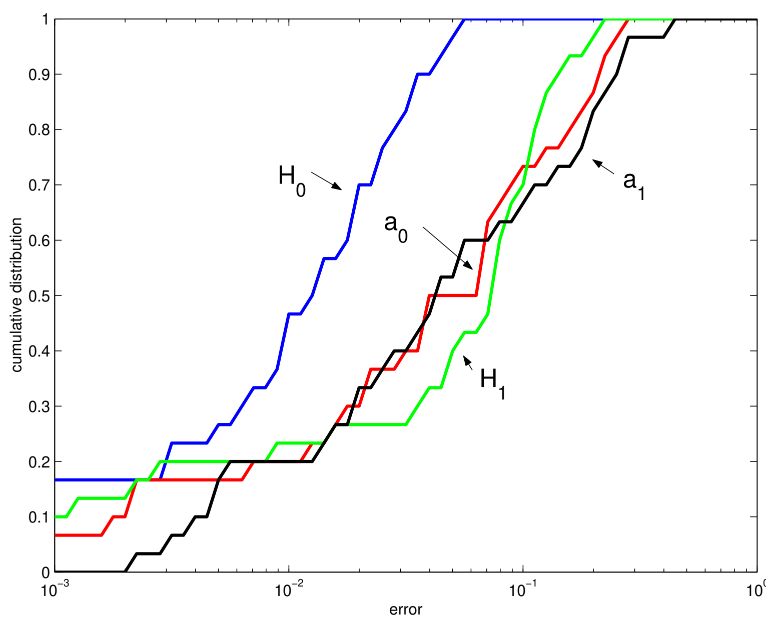
For $\rho = 0.7$, the difference between the a_0 and H_0 parameters at the ingress and egress node is quite small. This is to be expected since they represent the large time scale characteristics of the traffic. The reshaping that happens in a queue should happen over small time intervals. The difference for the a_1 and H_1 parameters is more significant. For H_1 there is a maximum error of 21%. Remember, since H_1 can only take values between 0.5 and 1, this could represent a significant change. The difference between a_1 at the ingress and egress can be as high as 63%. For $\rho = 0.95$ the difference is slightly larger.

While the parameter error may seem high, it is not immediately obvious if this error affects the queuing delay. When many flows are mixed in a queue, if the parameters of only a small number of flows are incorrect it may not affect the total queuing delay. To evaluate the impact of the parameter error on the queuing delay we perform the following test. For each queue in the network, we compute the queue length distribution using the parameters estimated for each flow at the input to the

[4]We increase the utilization from $\rho = 0.9$ used in the previous section to $\rho = 0.95$. In the previous section, the average traffic arrival rate on a link was 75 Mb/s, while in this simulation it is frequently above 150 Mb/s. For several links, utilization needs to reach 95% before the knee of the delay curve is reached.



(a) rho = 0.7



(b) rho = 0.95

Figure 3.12: Difference between the FBM parameters at the ingress and egress nodes

network. We also compute the queue length distribution using the actual parameters for each flow found in the simulation. We compare the 99.9th percentile of these delay distributions and find that the mean error is about 15%. There is one queue with an error of 200%, but for this queue, the magnitude of the delay is small. The delay computed using the input parameters is 1.2 ms, while the delay using the actual parameters is 0.4 ms. The queue with the next highest error has an error of 33%. These errors are significantly lower than the errors inherent in using the model (compare to the error in Figure 3.9). For the networks we consider, it is therefore reasonable to consider that a flow remains unchanged between ingress and egress and that the same model parameters may be used at each queue through which it passes.

3.4.2 End-to-end Delay Results

We now apply the procedure described at the beginning of this section to compute the end-to-end delay through the network shown in Figure 3.11. We compare the delays computed using the model with simulation results. As in the previous section we perform two simulations, one with $\rho = 0.7$ for all links and one with $\rho = 0.95$. We compute the percentage difference between the 99.9th percentile of delay distribution computed using the model and the 99.9th percentile of the delay distribution obtained in the simulation. We compute this difference for the delay between each ingress and egress node and plot the cumulative distribution in Figure 3.13. From this Figure, we see that between 80% of the ingress/egress node pairs the difference in the delay computed using the model and the delay in the simulation is less than 1. This holds

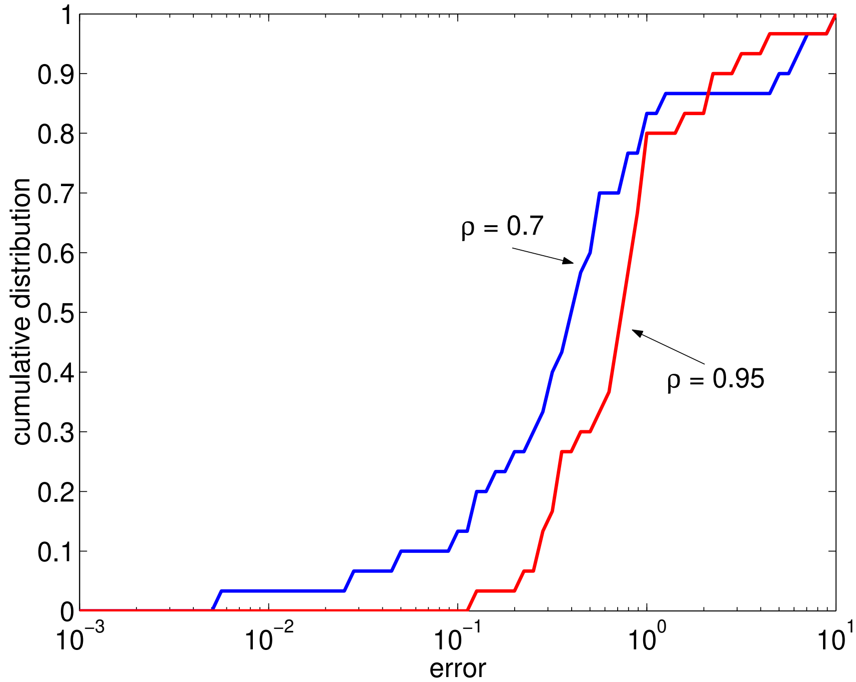


Figure 3.13: Difference between end-to-end delay predicted by the model and actual end-to-end delay

for both $\rho = 0.95$ and $\rho = 0.7$. These results are very similar to the error in the single hop delay shown in Figure 3.9. The most noticeable difference is that the error at $\rho = 0.95$ in the end-to-end delay case is quite a bit higher than the error for $\rho = 0.9$ in the single queue case. Recall at high utilization, the difference between the model and the simulation are diverging. As a result, the error at $\rho = 0.95$ will be higher than at $\rho = 0.9$. We conclude that the model provides accurate estimates for both the single hop delay as well as the end-to-end delay.

3.5 Summary

Using measurements of traffic collected on the Sprint IP network, we have found that when backbone traffic volume reaches 5 Mb/s to 50 Mb/s the marginal distribution of the traffic arrival process at time scales between 1 ms and 100 ms becomes Gaussian. This behavior is quite different from the results found in prior measurement studies which found very complex distributions at small time scales for traffic at lower levels of aggregation (1 Mb/s to 10 Mb/s). As a result of the Gaussian nature of backbone traffic, we are able to develop a parsimonious traffic model known as *two-scale Fractional Brownian* (FBM) motion. We derive the delay distribution for a queue fed by this model and develop a procedure to compute the end-to-end delay through a network carrying two-scale FBM traffic.

Chapter 4

Bandwidth Provisioning

4.1 Introduction

In Chapter 3 we developed an analytic model for backbone traffic known as *two-scale FBM*. In this chapter we demonstrate how this model is used to solve the bandwidth provisioning problem.

Consider a backbone network which is to be designed so that a constraint on the total end-to-end delay between any ingress and egress point is satisfied. This delay constraint is probabilistic and is specified as $P[d^{(i,j)} > D_{req}] < \epsilon$, that is the probability that the delay between node i and j exceeds D_{req} is less than ϵ. For the purposes of this discussion we consider that the propagation delay in the network is zero, and that $d^{(i,j)}$ and D_{req} represent the total queuing delay experienced. In actual networks, the propagation delay will of course be non-zero, but since this delay is a constant for all packets, it can simply be added to D_{req}. For example, when we consider an end-to-end D_{req} of 10 ms, this would correspond to a total end-to-end delay of 40 ms in a network with 30 ms of propagation delay. Alternatively, D_{req} can also be considered the maximum amount of delay jitter that is incurred in the network. In the example above the minimum delay a packet experiences would be 30 ms, while the maximum delay would be 40 ms.

There are several approaches which can be used to provision a network to meet such end-to-end delay requirements. One approach is to set a constraint on the total delay which can be incurred in a single queue in the network and determine the amount of bandwidth required on each link so that this delay constraint is met. For example, if D_{req} is 10 ms and the maximum number of hops between any ingress and egress point is 10, then each queue would be allowed to have 1 ms of queuing delay. Using the two-scale FBM model we can determine how much bandwidth is needed on a link so that the probability of a packet exceeding 1 ms delay on that link is less than ϵ. This approach is discussed in Section 4.2.

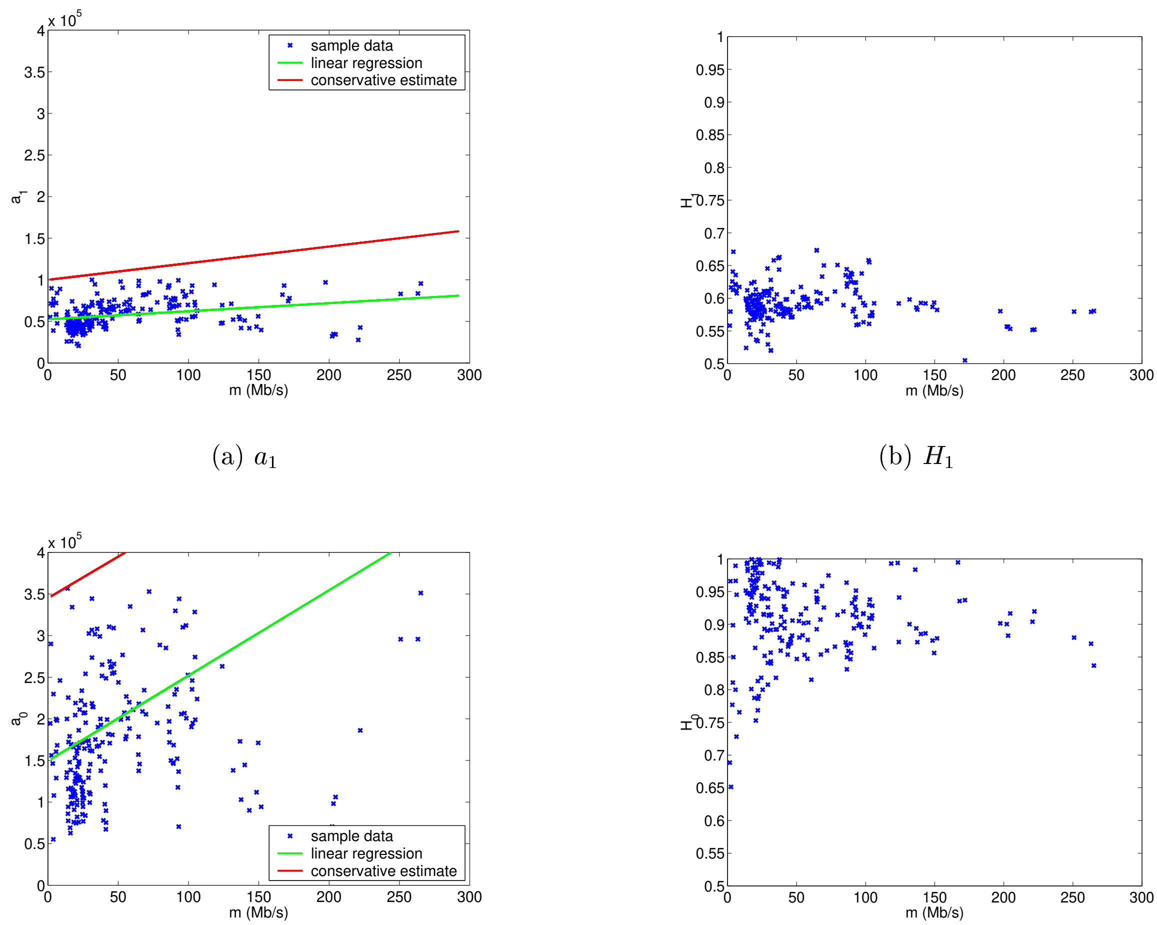
While the previous approach is a relatively straightforward method of bandwidth provisioning, it can be difficult to determine the appropriate delay constraint for a single queue. In the above example, we considered that the total end-to-end delay budget was divided evenly among all the hops along the end-to-end path. However, some hops may be over links with relatively low bandwidth (e.g. 155 Mb/s OC-3 links), while other hops may be over higher speed links (e.g. 10 Gb/s OC-192 links). Since the delays on the high bandwidth links will be lower, it may be inefficient to divide the total end-to-end delay budget evenly across all hops. Furthermore, when considering probabilistic delay requirements, the end-to-end delay is unlikely to be the sum of the delays at each hop. If only ϵ packets exceed the delay requirement at one hop, it is unlikely that the same ϵ packets exceed the delay requirement at the second hop. One could develop a strategy to set the delay requirements for each queue that takes into account all of these factors. However, an alternative solution is to develop an approach that computes the end-to-end delay along each path in the network and finds the amount of bandwidth required on each link so that the total end-to-end delay constraints are satisfied. This is a network optimization problem known as a Capacity Assignment (CA) problem and we present its solution in Section 4.3.

The two approaches described above are used to determine how much capacity is needed on each link in a network. However, installing new capacity can take between six months and two years. In some cases (e.g. when a new customer is added to the network), new traffic demands may be added to the network which result in the end-to-end delay constraints being violated. Since it is not possible to install bandwidth fast enough to meet these demands, the only alternative is to change the routing in the network to redistribute the traffic in a manner such that the end-to-end delay constraints are satisfied. In Section 4.4 we develop an algorithm to find a set of routes for the flows in a network such that a constraint on the total end-to-end delay is satisfied, if such a set of routes exists. For some traffic demands, it not possible to accommodate the additional traffic and still meet the end-to-end delay constraints.

4.2 Bandwidth Provisioning for a Single Link

A common question is: what is the maximum utilization at which a link can be operated while still meeting a particular delay requirement? The maximum achievable utilization on a link carrying two-scale FBM traffic can be computed directly using the delay distribution derived in Chapter 3. For a link of capacity C carrying traffic with an average arrival rate m, we can compute the delay distribution using Equation (3.6). To find the maximum achievable utilization, we need to find the maximum m which can be supported and still satisfy the delay constraint. To accomplish this we use binary search to find the maximum m.

The only remaining question is what are the four model parameters, (H_0, a_0, H_1, a_1) for traffic with an arrival rate of m. Using the measurements, we can make some projections. Figure 4.1 plots the model parameters for each traces against the traces' mean arrival rate, m. We see that the a_1 parameter (the variance at small time scales) exhibits a moderate increase with m. Using linear regression we find that $a_1 = 97m + 52562$ where m is the mean arrival rate in Mb/s. The parameter a_0, on the other hand, does not show as clear a trend. The best fit using linear regression finds the relationship $a_0 = 1027m + 149400$, but there is a wide range for the actual values of the parameters. The H_1 and H_0 parameters seem to be relatively stable across all values of m. For very small m (less than 20 Mb/s) we do see a wide range of values for H_0, but they converge to a value of 0.90 as m increases. This is to be expected since H_0 is a function of the connection size distribution. The connection size distribution should not change as m increases, as long as we are multiplexing



(a) a_1 (b) H_1

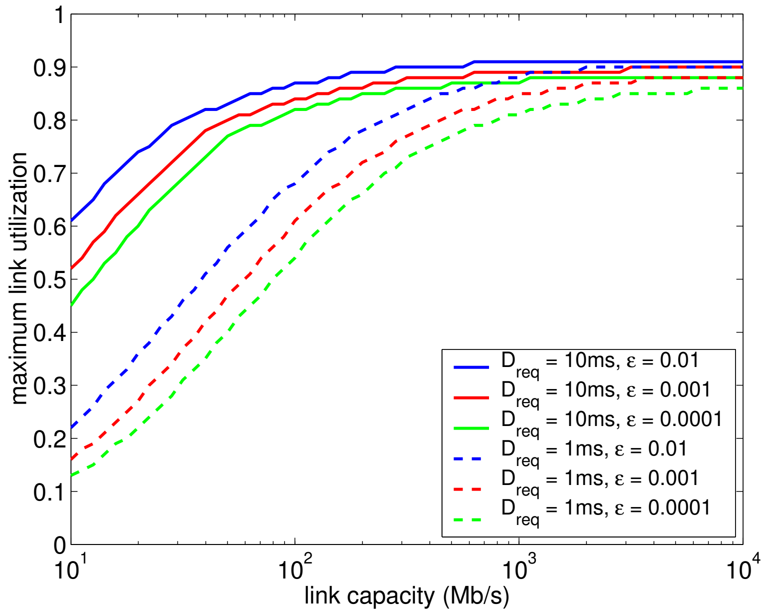
(c) a_0 (d) H_0

Figure 4.1: Two-scale FBM parameters for all traffic measurements

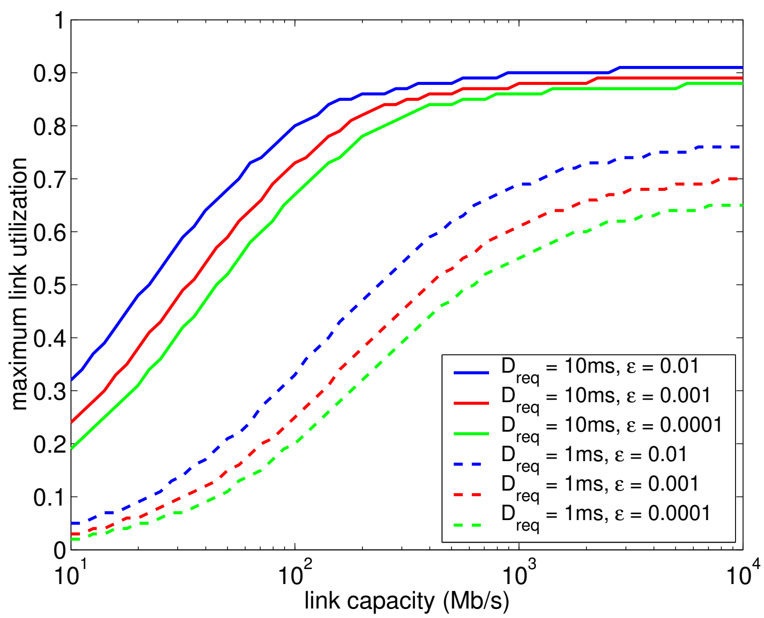
similar streams. H_1 converges to 0.59 in a similar fashion.

Since there is not a clear trend in the a_0 parameter, we consider two cases. First we consider the case of "average traffic". This corresponds to the linear regression lines shown in Figure 4.1. We also consider a "most variable" traffic approximation. This represents the traffic with the highest variability seen in all of our measurements and corresponds to the conservative estimate lines shown in Figure 4.1. For these two cases, we compute the maximum link utilization for a range of link capacities and plot the results in Figure 4.2. We show results for two different maximum delay requirements, $D_{req} = 10ms$ and $D_{req} = 1ms$ and three different delay percentiles, $\epsilon = 0.01$, $\epsilon = 0.001$, and $\epsilon = 0.0001$. We consider $D_{req} = 10ms$ as it is comparable to the 20 ms propagation delay for backbone networks. With $D_{req} = 10ms$, the sum of the queuing and propagation delay would be 30 ms. $D_{req} = 1ms$ may seem quite small relative to the propagation delay. However, we consider it to account for low jitter services. With $D_{req} = 1ms$, a provider could offer a service with an end-to-end delay of 20 ms and 1 ms of jitter.

Figure 4.2(a) shows the results assuming the traffic has characteristics similar to the average traffic while 4.2(b) shows the results for the most variable traffic. For links greater than 1 Gb/s, the typical bandwidth found in most backbone networks, link utilization can reach 80% to 90% for all but the most stringent SLAs. For such links, traffic differentiation will provide little bandwidth savings and relatively simple provisioning guidelines can be used (e.g. maintain utilization less than 80% on all links in the network). Between 10 Mb/s and 1 Gb/s the achievable link utilization



(a) average traffic



(b) most variable traffic

Figure 4.2: Maximum achievable link utilization

can vary quite a bit. For these links, simple guidelines are insufficient. It is important to know the exact model parameters for the traffic in the network and perform the queuing delay computations.

4.3 Capacity Assignment Problem

The results in the previous section can be used to provision a network if the delay requirement for a single queue can be derived from the end-to-end delay requirements which must be satisfied for the network. However, as we described at the beginning of this chapter, there are difficulties associated with determining the delay requirements for a single queue in the network. In this section we present an integrated procedure to determine the amount of bandwidth required on each link in the network so that an end-to-end delay constraint is satisfied.

Consider a network with fixed topology, fixed routing, and a known traffic demand between each ingress and egress point. For this network, the end-to-end delay between any ingress node and egress point must satisfy a probabilistic bound of the form $P[d^{(i,j)} > D_{req}] < \epsilon$. For this network, we are interested in finding the amount of bandwidth required on each link so that the end-to-end delay requirements are satisfied. A problem of this sort is referred to as a Capacity Assignment (CA) problem.

The CA problem has been solved for networks where the traffic flow between an ingress and egress node is modeled as a Poisson process with exponential service times and where the objective is to minimize the average delay [64]. Using Kleinrock's independence approximation and Jackson's theorem one can show that each ingress/egress flow remains Poisson throughout the network, and one can therefore derive expressions for the average queuing delay. Given an expression for the average delay, techniques such as Lagrangian relaxation are used to find the minimum cost network, where cost is the sum of the bandwidth on each link in the network. More complex cost functions which consider that a link has a fixed cost and a variable cost proportional to the bandwidth can also be used.

Our problem is different in three respects. First, many solutions to the CA problem assume a link can have any possible capacity (e.g. 100 Mb/s, 103 Mb/s, 106.34 Mb/s). In an actual network, only a discrete set of link capacities are available for each link (e.g. 155 Mb/s, 622 Mb/s, or 2.5 Gb/s). Restricting the link capacities to a discrete set transforms the problem into a complex integer programming problem in which must be solved using iterative heuristics. Second, we are interested in meeting a probabilistic delay requirement between each ingress and egress point rather than minimizing the average delay for a packet. Third, as demonstrated in Chapter 3, backbone traffic is quite different from Poisson traffic.

4.3.1 Problem Formulation

Let N be the set of nodes in the network, and L be the set of bidirectional links connecting the nodes. The capacity of each link $c_l, l \in L$, can be chosen from a finite set of possible link capacities C. The traffic which arrives to the network at node i and departs the network at node j is denoted by x_{ij} and is modeled by a two-scale FBM process with parameters $(m^{(i,j)}, H_1^{(i,j)}, a_1^{(i,j)}, H_0^{(i,j)}, a_0^{(i,j)})$. The flow x_{ij} follows path p_{ij} through the network. For this network, we would like to find the capacity assignment for all network links which satisfies the following properties:

Minimize the total network cost $M = \sum_{l \in L} c_l$

Subject to $P[d^{(i,j)} > D_{req}] < \epsilon$, $\forall i \in N, , j \in N, i \neq j$

For this problem, we consider the total cost of the network to be the sum of the individual link capacities, but the algorithm may be easily extended to handle more complex cost functions.

4.3.2 Capacity Assignment Algorithms

In most situations, it is possible to find the capacity allocation which results in the minimum network cost using a fairly simple algorithm. Consider a single link along an end-to-end delay path. The queuing delay at this link must be less than the total end-to-end delay allowed along the entire path. For a particular link, we can compute the minimum amount of bandwidth needed to satisfy this requirement. This process is repeated for every link, and the end-to-end delay along every path is computed. If the end-to-end delay constraints are satisfied, then we have found the capacity assignment which has the minimum network cost.

We find that in most situations, this procedure finds the minimum cost network. The reason for this is that the end-to-end delays distributions are computed by convolving the delay distributions at each hop (rather than summing the delay at each link as would be done to compute the average end-to-end delay). If the delay requirements are satisfied for each queue independently, then it is likely the convolution will not exceed the total end-to-end delay requirement.

However, this procedure is not guaranteed to work for all possible networks. When this procedure does not work iterative heuristics must be used to search for the minimum capacity assignment. Typically, the heuristic solutions operate as follows [7]. Each iteration of the heuristic begins with a *current capacity assignment*. A *trial capacity assignment* is generated by changing the capacity of one or more links in the current capacity assignment. If the trial capacity assignment satisfies the end-to-end delay constraints and the cost of the trial capacity assignment is lower, the trial capacity assignment replaces the current capacity assignment. The algorithm terminates when no further improvement is possible. Such an algorithm was proposed by Maruyama and Tang [72] to find a capacity assignment which minimizes cost and average packet delay. This algorithm is quite complex, but the basic approach is to generate new trial topologies by reducing the capacity of links which will result in the smallest increase in delay and increasing the capacity of those links which result in the greatest decrease in delay. The solution found by the Maruyama and Tang algorithm, as well as all other algorithms which only accept trial assignments with lower cost, may be improved by repeating the algorithm with a different initial solution. An alternative possibility is to allow the algorithm to accept some trial capacity assignments which have a higher cost than that of the current assignment. One such approach, known as simulated annealing has been proposed by Levi and Ersoy [69]. Learning automata [85] and genetic algorithm approaches [36] have also been used.

Algorithm 1 Simulated Annealing algorithm for Capacity Assignment Problem

initialization

k=0, P_0=0.90

choose initial feasible capacity assignment C_a

N = max(20, 2 * number of links)

for i = 1 to N

  create a neighbor of C_a called C_i such that:

    C_i is a feasible network

    $Cost(C_i) > Cost(C_k)$

$D_{mean} = \frac{1}{N} \sum_{i=1}^{N} Cost(C_i) - Cost(C_a)$

$t_k = t_0 = \frac{D_{mean}}{ln(1/P_0)}$

annealing

do

  $n = 0$

  do

    create neighbor of C_a called C_n

    if C_n is feasible

      if $Cost(C_n) < Cost(C_a)$ then $C_a = C_n$

      else if $e^{\frac{Cost(C_a)-Cost(C_n)}{t_k}} > random[0,1]$ then $C_a = C_n$

    $n = n + 1$

  until N_e neighbors are accepted or until n = 2 * number of links

  k=k + 1

  $t_k = \alpha * t_{k-1}$

until $Cost(C_a)$ is the same for N_s consecutive values of k

We chose to implement the simulated annealing heuristic as it has been shown to perform well when applied to the Capacity Assignment problem [69],[36]. Simulated annealing was originally developed by Kirkpatrick, Gelatt, and Vecchi [63], and has been applied to a wide range of optimization problems. The details of the algorithm are given in Algorithm 1.

The primary difference between Simulated Annealing and algorithms such as the Maruyama/Tang algorithm is that at each iteration of the algorithm, a trial capacity assignment with a higher cost than the current capacity assignment may be accepted with acceptance probability P. As the algorithm progresses, the acceptance probability is slowly decreased according to the *cooling schedule*. By accepting trial capacities with a higher cost, the algorithm avoids local minimum and converges to the optimal solution. In fact, with an optimal cooling schedule the algorithm is guaranteed to find the global minimum [1].

However, with an ideal cooling schedule, the algorithm would take an infinite amount of running time. In practice, cooling schedule parameters which achieve an acceptable tradeoff between the running time of the algorithm and the cost of the final solution must be found through experimentation. The four parameters of the cooling schedule are:

- P_0, the initial acceptance probability
- N_e, the length of each *epoch* - this is specified as the number of neighbors which must be accepted before transitioning to the new epoch
- α, the rate at which the acceptance probability is decreased between epochs

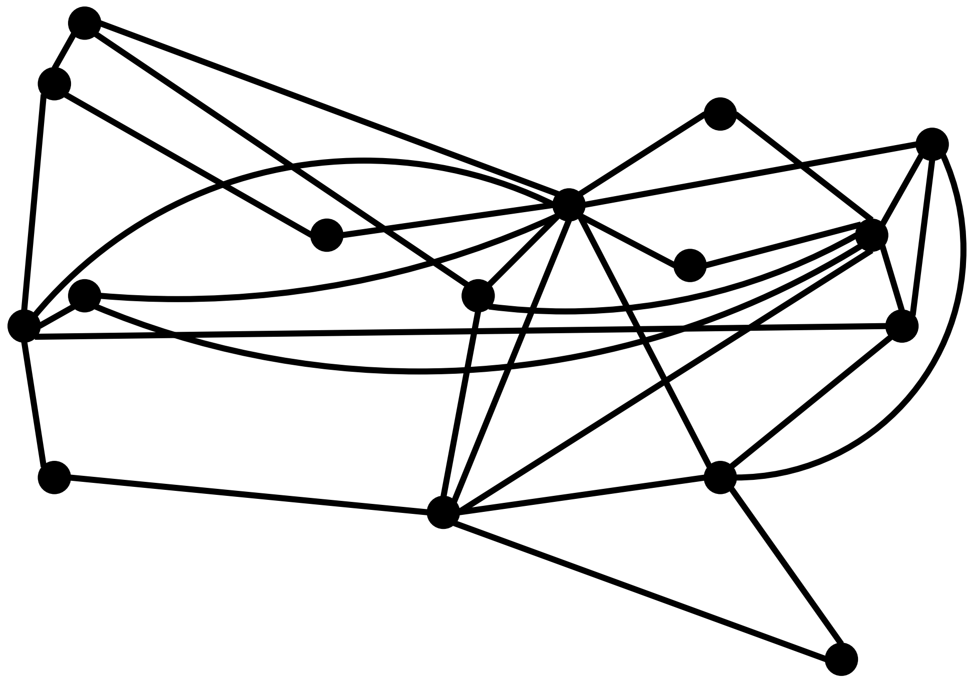


Figure 4.3: Sprint network topology

- N_s, the number of consecutive epochs which terminate with the same cost of the capacity assignment

Through experimentation we found using $P_0 = 0.9$, $N_e = 5$, $\alpha = 0.8$, and $N_s = 5$ resulted in the best performance.

4.3.3 Capacity Assignment for the Sprint IP Backbone

Using the algorithms described in the previous section, we can evaluate how much bandwidth is required in the Sprint IP backbone in order to meet various end-to-end delay constraints. By comparing the total cost of such a network to the total cost of a network which is designed just to support the average data rate of the traffic in the network without any delay requirements, we can determine if bandwidth provisioning

type 1	155 Mb/s
type 2	310 Mb/s
type 3	622 Mb/s
type 4	1.24 Gb/s
type 5	2.48 Gb/s
type 6	4.98 Gb/s
type 7	9.95 Gb/s
type 8	19.9 Gb/s

Table 4.1: Link types

is a feasible approach.

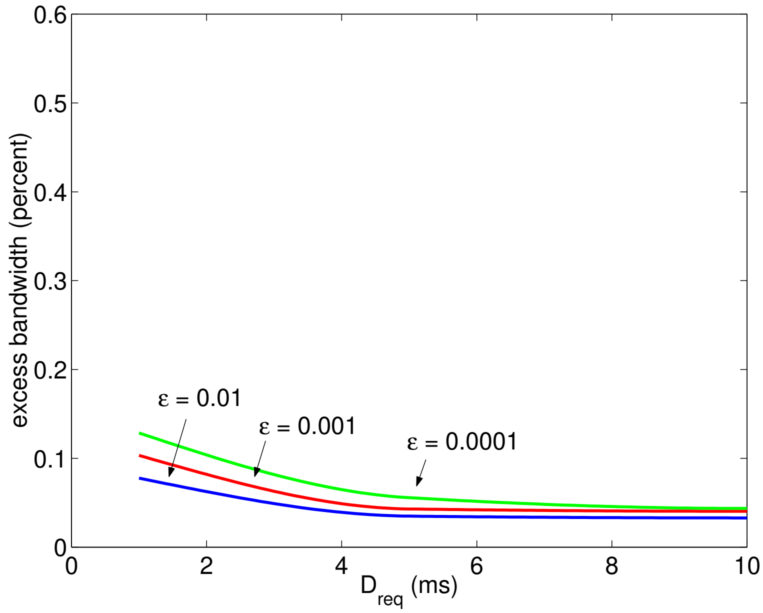
The topology of the Sprint network is shown in Figure 4.3. The topology and routing for the network are known, but we do not have measurements of the actual traffic demand matrix. To generate the traffic demand matrix, we use the approach outlined in [8]. We randomly classify 20% of the nodes in the network as "big" nodes, 40% as "medium" nodes, and 40% as "small" nodes. The mean traffic volume between an ingress i and egress j is selected from a Gaussian distribution with $mean^{(i,j)} = (size_i + size_j)/2$ where $size_{big} = 2.48$ Gb/s, $size_{medium} = 622$ Mb/s, and $size_{small} = 155$ Mb/s. The remaining four model parameters $(a_1^{(ij)}, H_1^{(ij)}, a_0^{(ij)}, H_0^{(ij)})$ are determined based on the mean arrival rate as described in Section 4.2. We consider both the average case and most variable traffic.

We generate five random node classifications and the resulting traffic matrices and find the capacity assignment which minimizes the network cost for a range of delay requirements. The set of possible link capacities is shown in Table 4.1. To find the capacity assignment we first use the simple algorithm. If the end-to-end delay constraints are not satisfied using the simple algorithm, we apply the Simulated

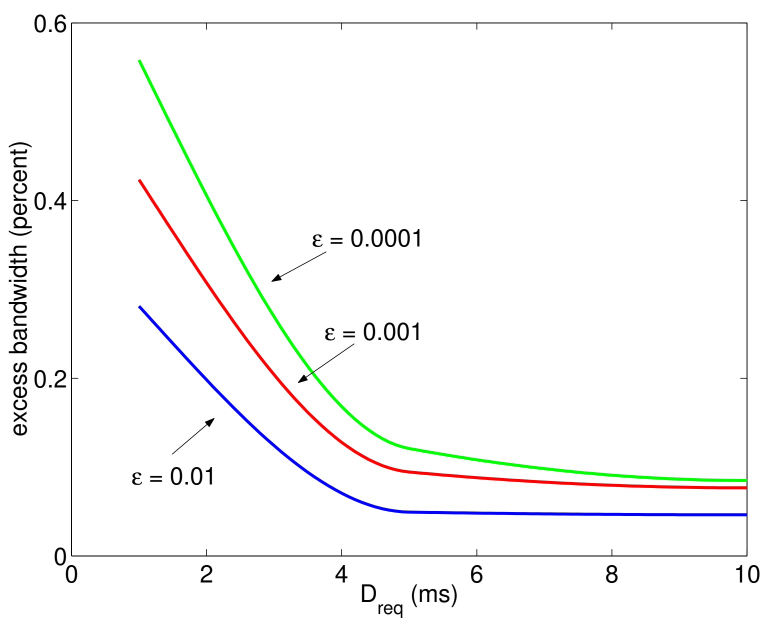
Annealing algorithm.

The feasibility of the bandwidth provisioning approach can be studied by comparing the bandwidth required to meet the end-to-end delay constraints with the minimum bandwidth required just to support the average rate of the traffic. We define the *excess bandwidth* as the percentage difference between the average rate of the traffic and the link capacity that is found in the solution to the capacity assignment problem: $bw_e = \sum_{l \in L} \frac{c_l - average\, traffic\, volume_l}{average\, traffic\, volume_l}$. The excess bandwidth is computed for each of the five traffic matrices, and the average is plotted for a range of delay requirements in Figure 4.4. We first consider a delay requirement in which only 1% of the packets exceed 10 ms queuing delay end-to-end through the network. If the network traffic corresponds to the "average traffic" from Section 4.2, the network needs only 3.2% excess bandwidth above the minimum bandwidth required to support the average traffic volume. If the network traffic is similar to the "most variable" traffic observed in all the measurements, only 4.6% excess bandwidth is required. Reducing the number of packets which can exceed the delay requirement to 0.01% results in only modest increases in the required bandwidth. These results are consistent with the results from Section 4.2 where we found that for links greater than 1 GB/s (which is the case for most of the links in the network we are considering), link utilization could reach 80% - 90% for delay requirements with $D_{req} = 10$ ms.

A delay requirement in which only 1% of the packets exceed 10 ms queuing delay end-to-end would be sufficient for even strict applications such as voice. Increasing the total end-to-end delay from 150 ms to 160 ms, for example, would not significantly



(a) average traffic



(b) most variable traffic

Figure 4.4: Excess bandwidth required to meet delay guarantees

degrade the performance of a voice call [70]. For such delay requirements, bandwidth provisioning seems to be an attractive solution. However, we are interested in understanding how low network queuing delays can be reduced before the bandwidth requirements begin to increase. Figure 4.4 therefore plots the excess bandwidth needed to support delay requirements ranging from 1 ms to 10 ms. For "average" network traffic, supporting delay requirements as low as 1 ms only requires up to 17% excess bandwidth. However, for the "most variable" network traffic, supporting end-to-end queuing delays of less than 3 ms requires significant amounts of excess bandwidth. In the extreme case of a delay requirement in which only 0.01% of the traffic exceeds 1 ms queuing delay end-to-end, the network would need almost 60% more bandwidth than the average traffic volume.

It is also interesting to compare these bandwidth requirements with the bandwidth that is installed based on current provisioning practices used by network providers today. Typically, network providers will establish a maximum link utilization threshold for each link and design the network so that the utilization is always below this threshold. These thresholds, however are set primarily based on intuition and experience. Consider the case where a network provider sets a maximum link utilization threshold of 50% for all links in the network (this corresponds to 100% more bandwidth than the average traffic volume). For the average traffic case and a delay requirement of $D_{req} = 5$ ms, $\epsilon = 0.001$, the provisioning approach described in this thesis only requires 8% more bandwidth than the average traffic volume, a 92% savings over the threshold approach.

4.4 Flow Assignment Problem

The previous two sections described methods for determining the amount of bandwidth needed to support a particular delay requirement. While these approaches are useful for long-term planning, there are situations in which traffic volume in the network may increase, and new capacity cannot be installed fast enough to meet the increase in demand. To accommodate these increases in demand, network operators reroute traffic in order to reduce the traffic volume on overloaded links. While it is not always possible to reroute traffic (alternative paths with sufficient capacity may not exist), in many cases moderate increases in traffic demand can be accommodated by finding alternate routes. In this section we develop an algorithm to find a set of routes for a given traffic demand so that an end-to-end delay constraint is satisfied.

Finding a set of routes to meet an end-to-end delay constraint can be formalized as a network optimization problem known as the Flow Assignment problem. This problem is very similar to the Capacity Assignment problem considered in the previous section. The difference between the two is that the Capacity Assignment problem considers the routes, p_{ij}, to be fixed and finds the capacity, C_l, for each link. The Flow Assignment problem considers the link capacities to be fixed and finds the route for each flow[1] . Furthermore, the objective of the Capacity Assignment problem is to minimize the total network cost, which is a function of the link capacities. In the Flow Assignment problem, the network capacities are known, so it is not appropriate to minimize the network cost. In fact, if one is simply trying to find a set of routes

[1] There is also a Capacity and Flow Assignment problem which simultaneously finds both the link capacities and the routes for each flow which minimize total network cost.

which will satisfy the end-to-end delay constraints an optimization step is not needed. However, there are several network parameters which it is desirable to optimize. One possible parameter is the average utilization of all links in the network. Such an approach, however, may result in some links which are very heavily loaded and some links which are very lightly loaded. To prevent such a situation, one can minimize the maximum utilization across all links. This will allow every link in the network to have some excess capacity which can be used to carry additional traffic. By optimizing the maximum link utilization, it is possible to have some network paths which are very long. For example, the maximum utilization may be reduced by allowing some flows to be routed from the East coast, to the West coast, and back to the East coast. To prevent such a situation from occurring, one can place constraints on the length of any path.

4.4.1 Problem Formulation

Let N be the set of nodes in the network, and L be the set of bidirectional links connecting the nodes. The capacity of each link, c_l, is known. The traffic which arrives to the network at node i and departs the network at node j is denoted by x_{ij} and is modeled by a two-scale FBM process with parameters $(m^{(i,j)}, H_1^{(i,j)}, a_1^{(i,j)}, H_0^{(i,j)}, a_0^{(i,j)})$. For this network, we would like to find the path for each flow, p_{ij}, which satisfies the following properties:

Minimize the weighted sum of the maximum link utilization and average path length: $M = \beta \max_{l \in L} utilization_l + (1-\beta) \sum_{i,j \in N} length_{p_{ij}} / |N|^2$

Subject to $P[d^{(i,j)} > D_{req}] < \epsilon$, $\forall i \in N, , j \in N, i \neq j$

In this problem definition, all of the traffic between node i and node j must follow the same path p_{ij}. However, it has been demonstrated that allowing the traffic to be split among a small number of alternate paths is beneficial [104]. In the example below, we therefore split each flow x_{ij} into four separate subflows $x_{ij}^k, k = (1, 2, 3, 4)$ and allow each subflow to follow a different path p_{ij}^k.

4.4.2 Flow Assignment Algorithm

The Flow Assignment algorithm is substantially more complex than the Capacity Assignment problem considered in the previous section. As a result, it is not possible to use a simple algorithm such as the single link algorithm developed for the CA problem. Instead, the only method for solving the Flow Assignment problem is through the use of heuristics. A simulated annealing algorithm similar to the one developed for the capacity assignment problem can be developed. This algorithm changes the path p_{ij}^k of a single flow at each iteration and searches for the path assignment which minimizes the maximum utilization and average path length.

However, the simulated annealing heuristic is much more complex than the simulated annealing algorithm for the capacity assignment problem. The algorithm for the capacity assignment problem changes a single link capacity at each iteration. This requires computing a new delay distribution for the link whose capacity was changed. The algorithm for the flow assignment problem changes a single path at each iteration. This requires computing new delay distributions for all of the links from which the

Algorithm 2 Algorithm to solve flow assignment problem

Set maximum utilization for each link, ρ_{max} based on delay requirement and results from Section 4.2

Simulated Annealing

initialization

k=0, P_0=0.90

choose initial flow assignment F_a

K = max(20, 2 * $|N|^2$)

for i = 1 to K

create a neighbor of F_a called F_i such that:

$\rho_l < \rho^l_{max}$ for all links l with flow assignment F_i

$M_i > M_k$

$D_{mean} = \frac{1}{K}\sum_{i=1}^{K} M_i - M_a$

$t_k = t_0 = \frac{D_{mean}}{ln(1/P_0)}$

annealing

do

$n = 0$

do

create neighbor of F_a called F_n

if $\rho_l < \rho^l_{max}$ for all links l with flow assignment F_n

if $M_n < M_a$ then $F_a = F_n$

else if $e^{\frac{M_a - M_n}{t_k}} > random[0,1]$ then $F_a = F_n$

$n = n + 1$

until N_e neighbors are accepted or until n = 2 * $|N|^2$

k=k + 1

$t_k = \alpha * t_{k-1}$

until M_a is the same for N_s consecutive values of k

compute end-to-end delay for all paths in the network

if end-to-end delay constraint is not satisfied, $\rho_{max} = 0.95\rho_{max}$ and repeat Simulated Annealing path was removed as well as new distributions for all the links to which the path was added. Furthermore, the flow assignment algorithm requires many more iterations than the capacity assignment algorithm. There are $4N^2$ flows in the network, and therefore $4N^2$ paths which must be selected from a large number of possible paths. If the capacity assignment algorithm were run on a full mesh network, there would be N^2 links whose capacity must be selected. However, backbone networks are not full mesh, and in the case of the Sprint network there are approximately $4N$ links, and each of these links had only eight possible capacities from which to choose.

As a result, implementing a simulated annealing algorithm which computes the exact end-to-end delay for every flow at each iteration of the algorithm is not feasible. Instead, for the simulated annealing portion of the algorithm, we only check that the average link utilization on each link is within the guidelines found in Section 4.2 for the particular delay requirement in which we are interested. Once the simulated annealing algorithm has terminated and found an assignment for each path, we check the exact end-to-end delays in order to determine if the requirements are satisfied. If not, we repeat the simulated annealing algorithm with slightly lower link utilization thresholds. The details of the algorithm are given in Algorithm 2.

4.4.3 Flow Assignment for Sprint IP Backbone

We are interested in determining how much extra traffic can be supported if the flow assignment algorithm is used to find routes for traffic rather than using shortest-path first routing, which is the approach used in most operational networks. To evaluate

	total network load (Mb/s)	percent increase
Initial traffic load	43143	-
Maximum load with shortest-path routing	43653	1%
Maximum load with flow assignment	53510	24%

Table 4.2: Total network load with and without dynamic routing

this, we begin with the set of flows x_{ij} and the link capacities c_l that were the solution of the capacity assignment problem presented in the previous section. This represents the capacity installed to meet a traffic demand of x_{ij} assuming each of the x_{ij}'s follows the shortest path between i and j. Due to the discrete nature of the link capacities, it is possible to modestly increase the total load in the network and still meet the end-to-end delay objectives even if all the flows still follow the shortest path. To determine how much the load may be increased, we multiply the mean rate of each flow by a factor $\alpha > 1$ and determine if the end-to-end delay constraint is satisfied with the increased traffic volume. We continue to increase α until the point at which the end-to-end delay constraint is violated. Next we consider that each flow x_{ij} can be divided into four subflows. The total amount of traffic in a flow is randomly divided among these subflows. We apply the flow assignment algorithm and determine how much further α may be increased before exceeding the end-to-end delay requirement.

Table 4.1 shows the results of this procedure. The total network load is simply the sum of the average rates of all flows x_{ij}. We present results corresponding to the case in which the "most variable" traffic considered in the capacity assignment section must meet an end-to-end delay requirement with $D_{req} = 10$ ms and $\epsilon = 0.01$. The table shows the average results for all five traffic matrices that were considered. From the table, we see that with shortest path routing, the total network load can only be increased by 1% and still satisfy the end-to-end delay constraints. However, using the flow assignment algorithm, we are able to increase the total network load by 24%. Beyond 24%, the delays in the network exceed the end-to-end delay constraint.

4.5 Summary

In this chapter we have developed procedures to evaluate the bandwidth required on each link in a network to support an end-to-end delay constraint. We find that for links with capacity greater than 1 Gb/s, utilization can reach 80%-90% and still meet all but the most stringent delay requirements. Similarly, for the network as a whole, only 5% - 15% excess bandwidth is needed to satisfy most end-to-end delays. In these situations, bandwidth provisioning is an attractive approach. However, to meet some very stringent delay requirements, link utilization on Gb/s links must be less than 60%. In these situations traffic differentiation will result in large bandwidth savings. Most of these bandwidth savings, however, can be realized without differentiation by simply allowing a few milliseconds of extra delay in the network.

Traffic differentiation does, however, provide benefits other than simply bandwidth savings. In particular, traffic differentiation may allow the network to meet the SLAs in the presence of unpredictable events (e.g., a sudden increase in traffic volume due to a flash crowd or link failure). While in some cases the flash crowd may be streaming the latest movie trailer or some other real-time data and traffic differentiation would not provide a benefit, there are many other types of unpredictable events for which differentiation does provide a benefit. To address such situations, we developed a flow assignment algorithm which can reroute traffic to take advantage of excess capacity available in the network. This algorithm allows the network load to be increased by nearly 25% before the end-to-end delay constraints are violated.

Chapter 5

Conclusion and Future Research

Interactive applications such as voice, audio, and video, as well as business applications such as Virtual Private Networks are becoming an increasingly important component of Internet traffic. Such applications have strict requirements on the total end-to-end delay which may be incurred in the network. There are two basic mechanism which may be used to meet the needs of these applications. One approach, known as traffic differentiation, is to give preferential treatment to latency sensitive traffic. While this approach can lead to efficient network design, there are costs associated with traffic differentiation. Traffic differentiation requires additional complexity in network routers, as well as added sophistication in network installation, management, and troubleshooting procedures. A second approach, known as bandwidth provisioning, is to provide enough resources (i.e. bandwidth) so that all traffic meets the most stringent delay requirement. This thesis investigated the bandwidth provisioning approach in the context of backbone networks to determine how much resources were required to meet various delay guarantees.

The amount of resources required is dependent on the characteristics of backbone network traffic. We therefore began by collecting and analyzing traffic measurements from the Sprint IP backbone, a commercial Tier-1 IP network. Chapter 2 described the architecture of the IPMON measurement facility which was designed to collect detailed packet-level traces from multiple locations in the Sprint network. These traces are used not only for the research described in this thesis, but also by a wide variety of research projects undertaken by the Sprint Advanced Technology Labs.

Chapter 2 also characterized backbone network traffic in terms of the protocols and applications which generate the traffic, and analyzed the characteristics of individual user flows in the network. Backbone network traffic is dominated by TCP traffic. On nearly all the links we monitor, over 90% of the traffic is TCP traffic. For early measurements which were collected in the summer of 2000, most of the traffic was generated by web applications. On a typical link 70% to 80% of the traffic would be web traffic. However, measurements collected in 2001 and 2002 demonstrate the peer-to-peer filesharing applications are becoming equally as important as web traffic. In fact, on some links, less than 20% of the traffic was web traffic and over 60% was filesharing traffic.

The most significant observation in terms of the bandwidth provisioning problem is that backbone network traffic is composed of a very large number of low rate users. In a one minute interval, a backbone link can carry between 10,000 and 300,000 individual user flows, depending on the link bandwidth. The average rate of a single user is typically less than 50 kb/s, and only 1% of the users exceed a rate of 500 kb/s. This is significantly smaller than the 155 Mb/s to 2.5 Gb/s capacity that is available on backbone links.

As a result of this large amount of aggregation, many statistical properties of backbone traffic follow Gaussian distribution. The property which is of interest when evaluating the queuing delay in the network is the traffic arrival process. Prior measurement studies have observed that the arrival process of network traffic, especially at small time scales, has quite complex distributions which cannot be fully characterized simply by the mean and variance. However, these results were based on measurements of traffic with an arrival rate between 200 kb/s and 10 Mb/s. Chapter 3 demonstrates that once the traffic volume has reached 50 Mb/s (and even for some cases for traffic volumes between 5 Mb/s and 50 Mb/s), there is sufficient aggregation that the traffic arrival process becomes Gaussian and *can* be fully characterized by its mean and variance alone. Chapter 3 develops a model known as *two-scale Fractional Brownian Motion* (FBM), whose arrival process matches that of backbone network traffic, and derives an expression for the delay distribution of a queue fed by two-scale FBM. In this chapter, it is also demonstrated that under realistic network scenarios, assumptions commonly made for Poisson traffic also hold for two-scale FBM and that it is therefore possible to evaluate the end-to-end delay through a network.

Analyzing a set of over 300 one-hour traffic measurements, Chapter 4 finds a set of two-scale FBM model parameters for "average" backbone traffic as well as the "most variable" backbone traffic. With these two models, the bandwidth provisioning approach can be evaluated for backbone networks. The bandwidth provisioning problem is addressed in three different ways. First, one could divide the total allowable end-to-end delay among the different links in the network. For example, if the total allowable end-to-end delay was 10 ms, and there were 10 links along the longest path in the network, each link could be allowed 1 ms of queuing delay. Chapter 4 determines the maximum utilization at which links of different capacity (e.g. 622 Mb/s OC-12 links, 2.5 Gb/s OC-48 links, etc.) can be operated while meeting such delay requirements. For links greater than 1 Gb/s, utilization can reach 80% - 90% and still meet all but the most stringent delay requirements.

However, dividing the end-to-end delay among different queues is not always a straightforward process. Therefore, a second approach to addressing the bandwidth provisioning is to consider the entire network, and determine the amount of bandwidth required on each link so that the total end-to-end delay constraints are satisfied. Chapter 4 develops two approaches to solving this problem. One approach is to find the bandwidth on each link so that the total end-to-end delay is not exceeded on that single link. In most situations, the resulting network will meet the total end-to-end delay requirements. However, in some cases this approach does not work, and an iterative search heuristic known as Simulated Annealing is used. When these procedures are applied to the Sprint network, we find that designing the network to support end-to-end queuing delays on the order of 5 ms to 10 ms requires only 5% - 15% excess bandwidth above the minimum bandwidth needed just to support the average traffic volume.

Installing new network capacity can take between six to 48 months. In some cases, traffic volume will increase and there is insufficient time to install new capacity. To accommodate such increases in traffic demand, traffic in the network can be rerouted in order to take advantage of excess capacity which is available on alternate paths. Chapter 4 develops an algorithm to find a set of routes for flows in a network so that the end-to-end delay constraints are not violated. Using this algorithm allows the network load to increase 24% beyond the load which can be supported using shortest-path first routing.

In summary, in the context of backbone networks, bandwidth provisioning is an attractive solution. As a result of the large amount of statistical multiplexing, as well as low packet transmission times, delays on backbone links remain less than several milliseconds until link utilization reaches 80% - 90%. As a result, across the entire network only 5% - 15% excess bandwidth is needed to support delay requirements which would be satisfactory to even stringent applications such as voice.

5.1 Future Work

This thesis developed an analytic model for backbone network traffic and applied this model to determine the amount of bandwidth needed to support various end-to-end delay requirements. There are several areas of interesting future research both in terms of the traffic model and in terms of supporting end-to-end delay guarantees.

In terms of the traffic model, there are two basic questions which are not addressed in this thesis. The first question is what causes the linear scaling behavior of the variance which is observed at small time scales. At large time scales, the linear scaling has been demonstrated to be the result of the distribution of user connection sizes [112]. The distribution of connection sizes determines how many users are expected to be transmitting over a given time interval. At small time scales, however, the traffic characteristics are dependent upon the manner in which users transmit individual packets, as well as the number of users that are active. For example, over a 1 second time interval, it can be approximated that all users with active connections were transmitting at their average rate for the 1 second interval. However, over a 10 ms interval, this approximation does not hold. Some active users may have transmitted packets over a particular 10 ms interval while other users with active connections may not have transmitted packets. The behavior of an individual user at these small time scales is heavily influenced by TCP.

A second area of future research related to the two-scale FBM model is to derive an expression for the combination of multiple two-scale FBM flows. In Chapter 3 we derive an expression for the delay distribution of a queue fed by multiple two-scale FBM flows, but we do not derive an explicit expression for the aggregate traffic stream. Such an expression would be significantly simplify the queuing delay analysis.

In terms of supporting end-to-end delay requirements there are also several interesting areas of future research. Traffic differentiation provides benefits other than simply bandwidth savings. In particular, traffic differentiation may allow the network to meet end-to-end delay guarantees in the presence of unpredictable events (e.g., a sudden increase in traffic volume due to a flash crowd or link failure). Link failures can be addressed using the provisioning algorithms developed in Chapter 4. These algorithms can be extended to handle reliability constraints, e.g., find the bandwidth required on each link so that the end-to-end delay constraints are not violated in the presence of N link failures. Unpredictable events such as flash crowds can be addressed in a similar manner using the provisioning algorithms. However, what is lacking is a model for such rare events.

5.2 Final Words

In this thesis we have seen that when network traffic is aggregated into large volumes, many of the complex characteristics of which were due to the behavior of a small number of users dissapear. As a result, it becomes easier to understand and model the network. Hopefully, as the Internet transitions from a period of extremely rapid growth to a more mature infrastructure this procedure will continue.

Bibliography

[1] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley, 1989.

[2] N. Anerousis, R. Caceres, N. Duffield, A. Feldmann, A. Greenberg, C. Kalmanek, P. Mishra, K.K. Ramakrishnan, and J. Rexford, "Using the AT&T Labs PacketScope for Internet measurement, design, and performance analysis," AT&T Labs Technical Report, Oct. 1997.

[3] J. Apisdorf, K.C. Claffy, K. Thompson, and R. Wilder, "OC3MON: Flexible, affordable, high performance statistics collection," in *Proceedings of INET '97*, Kuala Lumpur, Malaysia, June 1997.

[4] J.-M. Bardet and P. Bertrand, "Detecting abrupt change on the Hurst parameter of a multi-scale fractional Brownian motion with applications," in *New Directions in Time Series Analysis Workshop*, Luminy, France, Apr. 2001.

[5] A. Benassi and S. Deguy, "Multi-scale fractional Brownian motion: definition and identification," Tech. Rep. 83, LLAIC (Laboratoire de Logique, Algorithmique et Informatique de Clermont 1), France, Sep. 1999.

[6] Tim Berners-Lee, *Weaving the Web*, Harper Collins, 1999.

[7] D. Bertsekas and R. Gallager, *Data Networks, Second Edition*, Prentice Hall, 1992.

[8] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft, "Pop-level and access-link-level traffic dynamics in a Tier-1 POP," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, Nov. 2001.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," IETF Request For Comments, RFC 2475, Dec. 1998.

[10] A. Bouch, A. Kuchinsky, and N. Bhatti, "Quality is in the eye of the beholder: Meeting users's requirements for Internet quality of service," in *Proc. ACM Conference on Human Factors in Computing Systems*, the Hague, The Netherlands, Apr. 2000.

[11] J.-Y. Le Boudec and P. Thiran, *Network Calculus*, Springer Verlag Lecture Notes for Computer Science (LNCS) 2050, 2001.

[12] B. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview," IETF Request For Comments, RFC 1633, June 1994.

[13] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol RSVP - version 1 functional specification," IETF Request For Comments, RFC 2205, Sep. 1997.

[14] N. Brownlee, "Using NeTraMet for production traffic measurement," in *IFIP/IEEE International Symposium on Integrated Network Management*, Seattle, Washington, May. 2001.

[15] N. Brownlee, C. Mills, and G. Ruth, "Traffic flow measurement: Architecture," IETF Request For Comments, RFC 2722, Oct. 1999.

[16] R. Caceres, N. Duffield, A. Feldmann, J. Friedmann, A. Greenberg, R. Greer, T. Johnson, C. Kalmanek, B. Krishnamurthy, D. Lavelle, P. Mishra, K.K. Ramakrishnan, J. Rexford, F. True, and J. van der Merwe, "Measurement and analysis of IP network usage and behavior," *IEEE Communications*, vol. 38, no. 5, pp. 144–151, May. 2000.

[17] V.G. Cerf and R.E. Kahn, "A protocol for packet network interconnection," *IEEE Trans. on Communications*, vol. COM-22, no. 5, pp. 627–641, May. 1974.

[18] I.M. Chakravarti, R.G. Laha, and J. Roy, *Handbook of Methods of Applied Statistics, Volume I*, John Wiley and Sons, 1967.

[19] K.C. Claffy, G. Miller, and K. Thompson, "The nature of the beast: Recent traffic measurements from an Internet backbone," in *Proc. INET'98*, Geneva, Switzerland, July 1998.

[20] K.C. Claffy, G. Polyzos, and H.-W. Braun, "Traffic characteristics of the T1 NSFNET backbone," in *Proc. IEEE INFOCOM '93*, San Francisco, CA, Mar. 1993, pp. 885–892.

[21] J. Cleary, S. Donnelly, I. Graham, A. McGregor, and M. Pearson, "Design principles for accurate passive measurement," in *Proc. of Passive and Active Measurement: PAM-2000*, Hamilton, New Zealand, April 2000, pp. 1–8.

[22] K.G. Coffman and A.M. Odlyzko, "Growth of the Internet," in *Optical Fiber Telecommunications*, I.P. Kaminow and T. Li, Eds. 2002, pp. 17–56, Academic Press.

[23] D.R. Cox, "Long-range dependence: a review," in *Statistics: An Appraisal*, H.A. David and H.T. David, Eds. 1984, pp. 55–74, Iowa State University Press.

[24] M.E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes," *IEEE/ACM Trans. on Networking*, vol. 5, no. 6, pp. 835–846, Dec. 1997.

[25] M.E. Crovella and M.S. Taqqu, "Estimating the heavy tailed index from scaling properties," *Methodology and Computing in Applied Probability*, vol. 1, no. 1, pp. 55–79, Jan. 1999.

[26] R.L. Cruz, "A calculus for network delay, part I: Network elements in isolation," *IEEE Trans. on Info. Theory*, vol. 37, no. 1, pp. 114–131, Jan. 1991.

[27] R.L. Cruz, "A calculus for network delay, part II: Network analysis," *IEEE Trans. on Info. Theory*, vol. 37, no. 1, pp. 132–141, Jan. 1991.

[28] P.B. Danzig, S. Jamin, R. Caceres, D. Mitzel, and D. Estrin, "An empirical workload model for driving wide-area TCP/IP network simulations," *Journal of Interworking: Research and Experience*, vol. 3, no. 1, pp. 1–26, Mar. 1992.

[29] B. Davie, A. Charny, J.C.R. Bennett, K. Benson, J.-Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, "An expedited forwarding PHB (per-hop behavior)," IETF Request For Comments, RFC 2598, Mar. 2002.

[30] G. de Veciana, G. Kesidis, and J. Walrand, "Resource management in wide-area ATM networks using effective bandwidths," *IEEE J. on Selected Areas in Comm.*, vol. 13, no. 6, pp. 1081–1090, Aug. 1995.

[31] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Journal of Interworking: Research and Experience*, vol. 1, no. 1, pp. 3–26, Jan. 1990.

[32] N.G. Duffield and N. O'Connell, "Large deviations and overflow probabilities for the general single-server queue, with applications," in *Proc. Cambridge Phil. Soc.*, 1995, vol. 118, pp. 363–374.

[33] R.J. Edell, N. McKeown, and P. Varaiya, "Billing users and pricing for TCP," *IEEE J. on Selected Areas in Comm.*, vol. 13, no. 7, pp. 1162–1175, Sep. 1995.

[34] A. Erramilli, O. Narayan, A.L. Neidhardt, and I. Saniee, "Performance impacts of multi-scaling in wide area TCP/IP traffic," in *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, Mar. 2000, pp. 352–359.

[35] A. Erramilli, O. Narayan, and W. Willinger, "Experimental queuing analysis with long-range dependent packet traffic," *IEEE/ACM Trans. on Networking*, vol. 4, no. 2, pp. 209–223, Apr. 1996.

[36] C. Ersoy, A. Levi, and O. Gumrah, "Artifical intelligence search techniques for discrete link capacity assignment in prioritized multiservice network," *Int'l Journal of Computer System Science and Engineering*, vol. 15, no. 3, pp. 191–197, May. 2000.

[37] A. Feldmann, A.C. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *Proc. ACM SIGCOMM '99*, Cambridge, Massachusetts, Aug 1999, pp. 301–313.

[38] A. Feldmann, A.C. Gilbert, and W. Willinger, "Data networks as cascades: investigating the multifractal nature of Internet WAN traffic," in *Proc. ACM SIGCOMM '98*, Vancouver, Canada, Sep. 1998, pp. 42–55.

[39] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "NetScope: Traffic engineering for IP networks," *IEEE Network*, vol. 14, no. 2, pp. 11–19, March/April 2000.

[40] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: Methodology and experience," *IEEE/ACM Trans. on Networking*, vol. 9, no. 3, pp. 265–280, June 2001.

[41] D. Ferrari and D.C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE J. on Selected Areas in Comm.*, vol. 8, no. 3, pp. 544–551, Apr. 1990.

[42] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, D. Papagiannaki, and F. Tobagi, "Design and deployment of a passive monitoring infrastructure," in *Proc. of Passive and Active Measurement: PAM-2001*, Amsterdam, The Netherlands, Apr. 2001, pp. 12–22.

[43] C. Fraleigh, S. Moon, C. Diot, B. Lyles, and F. Tobagi, "Packet-level traffic measurements from a Tier-1 IP backbone," Tech. Rep. TR01-ATL-110101, Sprint Advanced Technology Labs, Nov. 2001.

[44] K. Frazer, "NSFNET : A partnership for high-speed networking," http://www.merit.edu/merit/archive/nsfnet/final.report/, 1995.

[45] R.J. Gibbens and P.J. Hunt, "Effective bandwidths for the multi-type UAS channel," *Queueing Systems*, vol. 9, no. 1-2, pp. 17–28, Sep. 1991.

[46] R.J. Gibbens, F.P. Kelly, and P.B. Key, "A decision-theoretic approach to call admission control in ATM networks," *IEEE J. on Selected Areas in Comm.*, vol. 13, no. 6, pp. 1101–1114, Aug. 1995.

[47] M. Grossglauser and J.-C. Bolot, "On the relevance of long-range dependence in network traffic," *IEEE/ACM Trans. on Networking*, vol. 7, no. 5, pp. 629–640, Oct. 1999.

[48] M. Grossglauser and D. Tse, "A framework for robust measurement-based admission control," *IEEE/ACM Trans. on Networking*, vol. 7, no. 3, pp. 293–309, June 1999.

[49] R. Guerin, H. Ahmadi, and N. Naghsineh, "Equivalent capacity and its application to high-speed networks," *IEEE J. on Selected Areas in Comm.*, vol. 9, no. 7, pp. 968–991, Sep. 1991.

[50] R. Guerin and V. Peris, "Quality-of-service in packet networks: Basic mechanisms and directions," *Computer Networks*, vol. 31, no. 3, pp. 169–189, Feb. 1999.

[51] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," IETF Request For Comments, RFC 2597, June 1999.

[52] G.K. Helder, "Customer evaluation of telephone circuits with delay," *Bell Systems Technical Journal*, vol. 45, no. 7, pp. 1157–1191, Sep. 1967.

[53] D.P. Heyman and T.V. Lakshman, "What are the implications of long-range dependence for VBR-video traffic engineering," *IEEE/ACM Trans. on Networking*, vol. 4, no. 3, pp. 301–317, June 1996.

[54] M. Horneffer, "Assessing Internet performance metrics using large-scale TCP SYN-based measurements," in *Proc. of Passive and Active Measurement: PAM-2000*, Hamilton, New Zealand, Apr. 2000.

[55] J.Y. Hui, "Resource allocation for broadband networks," *IEEE J. on Selected Areas in Comm.*, vol. 6, no. 9, pp. 1598–1608, Dec. 1988.

[56] et. al. J. Manchester, "IP over SONET/SDH," ANSI T1.X1.5/97-105 Technical Specification, 1997.

[57] V. Jacobson, C. Leres, and S. McCanne, "tcpdump," available via anonymous ftp to ftp.ee.lbl.gov, June 1989.

[58] S. Jamin, P. Danzig, S. Shenker, and L. Zhang, "A measurement based admission control algorithm for integrated services packet networks," *IEEE/ACM Trans. on Networking*, vol. 5, no. 1, pp. 56–70, Feb. 1997.

[59] S. Kalidindi and M. J. Zekauskas, "Surveyor: An infrastructure for Internet performance measurements," in *Proceedings of INET '99*, San Jose, CA, June 1999.

[60] F.P. Kelly, "Effective bandwidths at multi-class queues," *Queueing Systems*, vol. 9, no. 1-2, pp. 5–15, Sep. 1991.

[61] K. Keys, D. Moore, R. Koga, E. Lagache, M. Tesch, and K.C. Claffy, "The architecture of the CoralReef traffic monitoring software suite," in *Proc. of Passive and Active Measurement: PAM-2001*, Amsterdam, The Netherlands, Apr. 2001.

[62] J. Kimura, F. Tobagi, J.-M. Pulido, and P. Emstad, "Perceived quality and bandwidth characterization of layered MPEG-2 video encoding," in *Proc. SPIE*

Int'l Symposium on Voice, Video, and Data Communications, Boston, Sep. 1999.

[63] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May. 1983.

[64] L. Kleinrock, *Queueing Systems Volume 2: Computer Applications*, John Wiley and Sons, 1976.

[65] T.G. Kurtz, "Stochastic networks: Theory and applications," in *Limit theorems for workload input models*, F.P. Kelly, S. Zachary, and I. Ziedins, Eds. 1996, pp. 119–140, Oxford University Press.

[66] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, and S. Wolff, "A brief history of the internet," *Communications of the ACM*, vol. 40, no. 2, pp. 102–108, Feb. 1997.

[67] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. on Networking*, vol. 2, no. 1, pp. 1–15, Feb. 1994.

[68] W.E. Leland and D.V. Wilson, "High time-resolution measurement and analysis of LAN traffic: Implications for LAN interconnection," in *Proc. IEEE INFOCOM '91*, Bal Harbour, FL, Apr. 1991, pp. 1360–1366.

[69] A. Levi and C. Ersoy, "Discrete link capacity assignment in prioritized computer network: Two approaches," in *Proc. Ninth Int'l Symposium on Computer and Information Networks*, Antalya, Nov. 1994, pp. 408–415.

[70] A. Markopoulou and F. Tobagi, "Assessment of VoIP quality over internet backbones," in *Proc. IEEE INFOCOM 2002*, New York, New York, June 2002.

[71] H. Stele Martin, A. McGregor, and J.G. Cleary, "Analysis of Internet delay times," in *Proc. of Passive and Active Measurement: PAM-2000*, Hamilton, New Zealand, Apr. 2000.

[72] K. Maruyama and D.T. Tang, "Discrete link capacity assignment in communication networks," in *Proc. 3rd ICCC*, Toronto, Ontario, Aug. 1976, pp. 92–97.

[73] W. Matthews and L. Cottrel, "The PingER project: Active Internet performance monitoring for the HENP community," *IEEE Communications*, vol. 38, no. 5, pp. 130–136, May. 2000.

[74] S. McCreary and K.C. Claffy, "Trends in wide area IP traffic patterns: A view from Ames Internet Exchange," in *ITC Specialist Seminar*, Monterey, CA, May. 2000, pp. 1–1 – 1–12.

[75] A. McGregor, H.-W. Braun, and J. Brown, "The NLANR network analysis infrastructure," *IEEE Communications*, vol. 38, no. 5, pp. 122–128, May. 2000.

[76] D. McRobb, "cflowd: A traffic flow analysis tool for NetFlow," http://www.caida.org/tools/measurement/cflowd/, 1999.

[77] J. Micheel, S. Donnelly, and I. Graham, "Precision timestamping of network packets," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, Nov. 2001.

[78] D.L. Mills, "Network time protocol (version 3) specification, implementation, and analysis," IETF Request For Comments, RFC 1305, Mar. 1992.

[79] D.L. Mills, "A kernel model for precision timekeeping," IETF Request For Comments, RFC 1589, Mar. 1994.

[80] D.L. Mills, "Clock discipline algorithms for the network time protocol version 4," Tech. Rep. 97-3-3, University of Delaware, Mar. 1997.

[81] "NetFlow services and applications," http://www.cisco.com/warp/public/cc/-pd/iosw/ioft/neflct/tech/napps_wp.htm, Cisco white paper.

[82] I. Norros, "A storage model with self-similar input," *Queuing Systems*, vol. 16, pp. 387–396, 1994.

[83] I. Norros, "On the use of Fractional Brownian Motion in the theory of connectionless networks," *IEEE J. on Selected Areas in Comm.*, vol. 13, no. 6, pp. 953–962, Aug 1995.

[84] W. Noureddine and F. Tobagi, "Improving the performance of interactive TCP applications using service differentiation," in *Proc. IEEE INFOCOM 2002*, New York, New York, June 2002.

[85] B.J. Oommen and T.D. Roberts, "Continuous learning automata solutions to the capacity assignment problem," *IEEE Trans. on Computers*, vol. 49, no. 6, pp. 608–620, June 2000.

[86] D. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot, "Analysis of measured single-hop delay from an operational backbone network," in *Proc. IEEE INFOCOM 2002*, New York, New York, June 2002.

[87] A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Trans. on Networking*, vol. 1, no. 3, pp. 344–357, June 1993.

[88] A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case," *IEEE/ACM Trans. on Networking*, vol. 2, no. 2, pp. 137–150, Apr. 1994.

[89] V. Paxson, A. Adams, and M. Mathis, "Experiences with NIMI," in *Proc. of Passive and Active Measurement: PAM-2000*, Hamilton, New Zealand, Apr. 2000.

[90] V. Paxson and S. Floyd, "Wide-area traffic: The failure of Poisson modeling," *IEEE/ACM Trans. on Networking*, vol. 3, no. 3, pp. 226–244, June 1995.

[91] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, "An architecture for large-scale Internet measurement," *IEEE Communications*, vol. 36, no. 8, pp. 48–54, Aug. 1998.

[92] Vern Paxson, “Empirically-derived analytic models of wide-area TCP connections,” *IEEE/ACM Trans. on Networking*, vol. 2, no. 4, pp. 316–336, August 1994.

[93] Vern Paxson, “End-to-end routing behavior in the Internet,” *IEEE/ACM Trans. on Networking*, vol. 5, no. 5, pp. 601–615, Oct. 1997.

[94] Vern Paxson, “End-to-end Internet packet dynamics,” *IEEE/ACM Trans. on Networking*, vol. 7, no. 3, pp. 277–292, June 1999.

[95] J. Postel, “Internet protocol - DARPA Internet program protocol specification,” IETF Request For Comments, RFC 791, Sep. 1981.

[96] J. Postel, “Transmission control protocol - DARPA Internet program protocol specification,” IETF Request For Comments, RFC 793, Sep. 1981.

[97] V.J. Ribeiro, R.H. Riedi, M.S. Crouse, and R.G. Baraniuk, “Multiscale queuing analysis of long-range dependent network traffic,” in *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, Mar. 2000, pp. 1026–1035.

[98] R.H. Riedi, M.S. Crouse, V.J. Ribeiro, and R.G. Baraniuk, “A multifractal wavelet model with application to network traffic,” *IEEE Trans. on Info. Theory*, vol. 45, no. 4, pp. 992–1018, May. 1999.

[99] J. Rosenberg, L. Qui, and H. Schulzrinne, “Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet,” in *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, Mar. 2000, pp. 1705–1714.

[100] B.K. Ryu and A. Elwalid, "The importance of long-range dependence of VBR video traffic in ATM traffic engineering," in *Proc. ACM SIGCOMM '96*, San Francisco, California, August 1996, pp. 3–14.

[101] G. Samorodnitsky and M.S. Taqqu, *Stable non-Gaussian Random Processes*, Chapman & Hall, 1994.

[102] W. Simpson, "PPP in HDLC framing," IETF Request For Comments, RFC 1549, Dec. 1993.

[103] W. Simpson, "PPP over SONET/SDH," IETF Request For Comments, RFC 1619, May. 1994.

[104] A. Sridharan, S. Bhattacharyya, C. Diot, R. Guerin, J. Jetcheva, and N. Taft, "On the impact of aggregation on the performance of traffic aware routing," in *Proc. 17th Int'l Teletraffic Congress*, Salvador do Bahia, Brazil, Sep. 2001.

[105] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in *Proc. ACM SIGCOMM '99*, Boston, Sep. 1999, pp. 81–94.

[106] L. Subramanian, S. Agarwal, J. Rexford, and R.H. Katz, "Characterizing the Internet hierarchy from multiple vantage points," in *Proc. IEEE INFOCOM 2002*, New York, New York, June 2002.

[107] M. Taqqu, V. Teverovsky, and W. Willinger, "Estimators for long-range dependence: An empirical study," *Fractals*, vol. 3, no. 4, pp. 785–798, 1995.

[108] K. Thompson, G. Miller, and R. Wilder, "Wide area Internet traffic patterns and characteristics," *IEEE Network*, vol. 11, no. 6, pp. 10–23, Nov. 1997.

[109] H. Uijterwall and D. Karrenberg, "Internet delay measurements using test traffic: First results," in *Proc. of SANE '98*, November 1998.

[110] D. Veitch and P. Abry, "A wavelet based joint estimator for the parameters of long-range dependence," *IEEE Trans. on Info. Theory*, vol. 45, no. 3, pp. 878–897, Apr. 1999.

[111] I. Wakeman, D. Lewis, and J. Crowcroft, "Traffic analysis of trans-atlantic traffic," *Computer Communications*, vol. 36, no. 1, pp. 376–388, June 1993.

[112] W. Willinger, M.S. Taqqu, R. Sherman, and D.V. Wilson, "Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level," *IEEE/ACM Trans. on Networking*, vol. 5, no. 1, pp. 71–86, Feb. 1997.

[113] D. Wischik, "The output of a switch, or, effective bandwidths for networks," *Queueing Systems*, vol. 32, pp. 383–396, 1999.

[114] R. Zakon, "Hobbes' Internet timeline," IETF Request For Comments, RFC 2235, Nov. 1997.

[115] H. Zhang, "Service disciplines for guaranteed performance service in packet-switched networks," *Proc. of the IEEE*, vol. 83, no. 10, pp. 1375–1396, Oct. 1995.